

new/exception_lists/cstyle

1

```
*****
45302 Sat Jul 19 19:24:26 2014
new/exception_lists/cstyle
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1  usr/src/cmd/krb5/kadmin/cli/kadmin_ct.c
2  usr/src/cmd/krb5/kadmin/cli/kadmin.c
3  usr/src/cmd/krb5/kadmin/cli/kadmin.h
4  usr/src/cmd/krb5/kadmin/cli/keytab.c
5  usr/src/cmd/krb5/kadmin/cli/ss_wrapper.c
6  usr/src/cmd/krb5/kadmin/dbutil/dump.c
7  usr/src/cmd/krb5/kadmin/dbutil/import_err.h
8  usr/src/cmd/krb5/kadmin/dbutil/kadm5_create.c
9  usr/src/cmd/krb5/kadmin/dbutil/kdb5_create.c
10 usr/src/cmd/krb5/kadmin/dbutil/kdb5_destroy.c
11 usr/src/cmd/krb5/kadmin/dbutil/kdb5_stash.c
12 usr/src/cmd/krb5/kadmin/dbutil/kdb5_util.c
13 usr/src/cmd/krb5/kadmin/dbutil/kdb5_util.h
14 usr/src/cmd/krb5/kadmin/dbutil/nstrtok.h
15 usr/src/cmd/krb5/kadmin/dbutil/ovload.c
16 usr/src/cmd/krb5/kadmin/dbutil/string_table.c
17 usr/src/cmd/krb5/kadmin/dbutil/string_table.h
18 usr/src/cmd/krb5/kadmin/dbutil/strtok.c
19 usr/src/cmd/krb5/kadmin/dbutil/util.c
20 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd_strings.h
21 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd.c
22 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd.h
23 usr/src/cmd/krb5/kadmin/kpasswd/tty_kpasswd.c
24 usr/src/cmd/krb5/kadmin/ktutil/ktutil_ct.c
25 usr/src/cmd/krb5/kadmin/ktutil/ktutil_funcs.c
26 usr/src/cmd/krb5/kadmin/ktutil/ktutil.c
27 usr/src/cmd/krb5/kadmin/ktutil/ktutil.h
28 usr/src/cmd/krb5/kadmin/server/kadm_rpc_svc.c
29 usr/src/cmd/krb5/kadmin/server/misc.c
30 usr/src/cmd/krb5/kadmin/server/misc.h
31 usr/src/cmd/krb5/kadmin/server/ovsec_kadmd.c
32 usr/src/cmd/krb5/kadmin/server/server_glue_v1.c
33 usr/src/cmd/krb5/kadmin/server/server_stubs.c
34 usr/src/cmd/krb5/kdestroy/kdestroy.c
35 usr/src/cmd/krb5/kinit/kinit.c
36 usr/src/cmd/krb5/klist/klist.c
37 usr/src/cmd/krb5/krb5kdc/dispatch.c
38 usr/src/cmd/krb5/krb5kdc/do_as_req.c
39 usr/src/cmd/krb5/krb5kdc/do_tgs_req.c
40 usr/src/cmd/krb5/krb5kdc/extern.c
41 usr/src/cmd/krb5/krb5kdc/extern.h
42 usr/src/cmd/krb5/krb5kdc/kdc_preauth.c
43 usr/src/cmd/krb5/krb5kdc/kdc_util.c
44 usr/src/cmd/krb5/krb5kdc/kdc_util.h
45 usr/src/cmd/krb5/krb5kdc/main.c
46 usr/src/cmd/krb5/krb5kdc/network.c
47 usr/src/cmd/krb5/krb5kdc/policy.c
48 usr/src/cmd/krb5/krb5kdc/policy.h
49 usr/src/cmd/krb5/krb5kdc/replay.c
50 usr/src/cmd/krb5/krb5kdc/sock2p.c
51 usr/src/cmd/krb5/ldap_util/kdb5_ldap_list.c
52 usr/src/cmd/krb5/ldap_util/kdb5_ldap_list.h
53 usr/src/cmd/krb5/ldap_util/kdb5_ldap_policy.c
54 usr/src/cmd/krb5/ldap_util/kdb5_ldap_policy.h
55 usr/src/cmd/krb5/ldap_util/kdb5_ldap_realm.c
56 usr/src/cmd/krb5/ldap_util/kdb5_ldap_realm.h
```

new/exception_lists/cstyle

2

```
57  usr/src/cmd/krb5/ldap_util/kdb5_ldap_services.c
58  usr/src/cmd/krb5/ldap_util/kdb5_ldap_services.h
59  usr/src/cmd/krb5/ldap_util/kdb5_ldap_util.c
60  usr/src/cmd/krb5/ldap_util/kdb5_ldap_util.h
61  usr/src/cmd/krb5/slave/kprop.c
62  usr/src/cmd/krb5/slave/kprop.h
63  usr/src/cmd/krb5/slave/kpropd.c
64  usr/src/cmd/mandoc/arch.c
65  usr/src/cmd/mandoc/att.c
66  usr/src/cmd/mandoc/chars.c
67  usr/src/cmd/mandoc/config.h
68  usr/src/cmd/mandoc/eqn.c
69  usr/src/cmd/mandoc/eqn_html.c
70  usr/src/cmd/mandoc/eqn_term.c
71  usr/src/cmd/mandoc/html.c
72  usr/src/cmd/mandoc/html.h
73  usr/src/cmd/mandoc/lib.c
74  usr/src/cmd/mandoc/libman.h
75  usr/src/cmd/mandoc/libmandoc.h
76  usr/src/cmd/mandoc/libmdoc.h
77  usr/src/cmd/mandoc/libroff.h
78  usr/src/cmd/mandoc/main.c
79  usr/src/cmd/mandoc/main.h
80  usr/src/cmd/mandoc/man.c
81  usr/src/cmd/mandoc/man.h
82  usr/src/cmd/mandoc/man_hash.c
83  usr/src/cmd/mandoc/man_html.c
84  usr/src/cmd/mandoc/man_macro.c
85  usr/src/cmd/mandoc/man_term.c
86  usr/src/cmd/mandoc/man_validate.c
87  usr/src/cmd/mandoc/mandoc.c
88  usr/src/cmd/mandoc/mandoc.h
89  usr/src/cmd/mandoc/mdoc.c
90  usr/src/cmd/mandoc/mdoc.h
91  usr/src/cmd/mandoc/mdoc_argv.c
92  usr/src/cmd/mandoc/mdoc_hash.c
93  usr/src/cmd/mandoc/mdoc_html.c
94  usr/src/cmd/mandoc/mdoc_macro.c
95  usr/src/cmd/mandoc/mdoc_man.c
96  usr/src/cmd/mandoc/mdoc_term.c
97  usr/src/cmd/mandoc/mdoc_validate.c
98  usr/src/cmd/mandoc/msec.c
99  usr/src/cmd/mandoc/out.c
100 usr/src/cmd/mandoc/out.h
101 usr/src/cmd/mandoc/preconv.c
102 usr/src/cmd/mandoc/read.c
103 usr/src/cmd/mandoc/roff.c
104 usr/src/cmd/mandoc/st.c
105 usr/src/cmd/mandoc/tbl.c
106 usr/src/cmd/mandoc/tbl_data.c
107 usr/src/cmd/mandoc/tbl_html.c
108 usr/src/cmd/mandoc/tbl_layout.c
109 usr/src/cmd/mandoc/tbl_opts.c
110 usr/src/cmd/mandoc/tbl_term.c
111 usr/src/cmd/mandoc/term.c
112 usr/src/cmd/mandoc/term.h
113 usr/src/cmd/mandoc/term_ascii.c
114 usr/src/cmd/mandoc/term_ps.c
115 usr/src/cmd/mandoc/tree.c
116 usr/src/cmd/mandoc/vol.c
117  usr/src/common/bzip2/bzlib.h
118  usr/src/common/bzip2/crcrtable.c
119  usr/src/common/bzip2/randtable.c
120  usr/src/common/bzip2/blocksort.c
121  usr/src/common/bzip2/compress.c
122  usr/src/common/bzip2/bzlib.c
```

new/exception_lists/cstyle

```

123 usr/src/common/bzip2/decompress.c
124 usr/src/common/bzip2/bzlib_private.h
125 usr/src/common/bzip2/huffman.c
126 usr/src/common/openssl/crypto/krb5/krb5_asn.c
127 usr/src/common/openssl/crypto/krb5/krb5_asn.h
128 usr/src/lib/gss_mechs/mech_krb5/crypto/aes/aes_s2k.c
129 usr/src/lib/gss_mechs/mech_krb5/crypto/cksumtype_to_string.c
130 usr/src/lib/gss_mechs/mech_krb5/crypto/coll_proof_cksum.c
131 usr/src/lib/gss_mechs/mech_krb5/crypto/crc32/crc.c
132 usr/src/lib/gss_mechs/mech_krb5/crypto/des/afsstring2key.c
133 usr/src/lib/gss_mechs/mech_krb5/crypto/des/string2key.c
134 usr/src/lib/gss_mechs/mech_krb5/crypto/dk/stringtokey.c
135 usr/src/lib/gss_mechs/mech_krb5/crypto/enctype_compare.c
136 usr/src/lib/gss_mechs/mech_krb5/crypto/enctype_to_string.c
137 usr/src/lib/gss_mechs/mech_krb5/crypto/hash_provider/hash_md5.c
138 usr/src/lib/gss_mechs/mech_krb5/crypto/hash_provider/hash_shal.c
139 usr/src/lib/gss_mechs/mech_krb5/crypto/keyed_checksum_types.c
140 usr/src/lib/gss_mechs/mech_krb5/crypto/keyed_cksum.c
141 usr/src/lib/gss_mechs/mech_krb5/crypto/keyhash_provider/hmac_md5.c
142 usr/src/lib/gss_mechs/mech_krb5/crypto/keyhash_provider/k5_md5des.c
143 usr/src/lib/gss_mechs/mech_krb5/crypto/keylengths.c
144 usr/src/lib/gss_mechs/mech_krb5/crypto/make_random_key.c
145 usr/src/lib/gss_mechs/mech_krb5/crypto/md4/md4.c
146 usr/src/lib/gss_mechs/mech_krb5/crypto/old_api_glue.c
147 usr/src/lib/gss_mechs/mech_krb5/crypto/old/des_stringtokey.c
148 usr/src/lib/gss_mechs/mech_krb5/crypto/pbkdf2.c
149 usr/src/lib/gss_mechs/mech_krb5/crypto/random_to_key.c
150 usr/src/lib/gss_mechs/mech_krb5/crypto/state.c
151 usr/src/lib/gss_mechs/mech_krb5/crypto/string_to_cksumtype.c
152 usr/src/lib/gss_mechs/mech_krb5/crypto/string_to_enctype.c
153 usr/src/lib/gss_mechs/mech_krb5/crypto/string_to_key.c
154 usr/src/lib/gss_mechs/mech_krb5/crypto/valid_cksumtype.c
155 usr/src/lib/gss_mechs/mech_krb5/crypto/valid_enctype.c
156 usr/src/lib/gss_mechs/mech_krb5/et/com_err.c
157 usr/src/lib/gss_mechs/mech_krb5/et/error_message.c
158 usr/src/lib/gss_mechs/mech_krb5/et/error_table.h
159 usr/src/lib/gss_mechs/mech_krb5/et/internal.h
160 usr/src/lib/gss_mechs/mech_krb5/et/mit-sipb-copyright.h
161 usr/src/lib/gss_mechs/mech_krb5/include/cache-addrinfo.h
162 usr/src/lib/gss_mechs/mech_krb5/include/cm.h
163 usr/src/lib/gss_mechs/mech_krb5/include/com_err.h
164 usr/src/lib/gss_mechs/mech_krb5/include/db-config.h
165 usr/src/lib/gss_mechs/mech_krb5/include/db.h
166 usr/src/lib/gss_mechs/mech_krb5/include/fake-addrinfo.h
167 usr/src/lib/gss_mechs/mech_krb5/include/foreachaddr.h
168 usr/src/lib/gss_mechs/mech_krb5/include/k5-int-pkinit.h
169 usr/src/lib/gss_mechs/mech_krb5/include/k5-utf8.h
170 usr/src/lib/gss_mechs/mech_krb5/include/kdb_kt.h
171 usr/src/lib/gss_mechs/mech_krb5/include/krb5_libinit.h
172 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_defs.h
173 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_proto.h
174 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm.h
175 usr/src/lib/gss_mechs/mech_krb5/include/krb5/copyright.h
176 usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-err.h
177 usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-plugin.h
178 usr/src/lib/gss_mechs/mech_krb5/include/krb5/kdb_dbc.h
179 usr/src/lib/gss_mechs/mech_krb5/include/krb5/kdb.h
180 usr/src/lib/gss_mechs/mech_krb5/include/locate_plugin.h
181 usr/src/lib/gss_mechs/mech_krb5/include/osconf.h
182 usr/src/lib/gss_mechs/mech_krb5/include/port-sockets.h
183 usr/src/lib/gss_mechs/mech_krb5/include/preauth_plugin.h
184 usr/src/lib/gss_mechs/mech_krb5/include/socket-utils.h
185 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.c
186 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.h
187 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.c
188 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.h

```

3

new/exception_lists/cstyle

```

189 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.c
190 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.h
191 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_decode.c
192 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_decode.h
193 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_encode.c
194 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_encode.h
195 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.c
196 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.h
197 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.c
198 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.h
199 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1buf.h
200 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krb5_decode.c
201 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krb5_encode.c
202 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krbasn1.h
203 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/ldap_key_seq.c
204 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc_file.c
205 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc_memory.c
206 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc_retr.c
207 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc-int.h
208 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccbase.c
209 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cccopy.c
210 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccdefault.c
211 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccdefops.c
212 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccfns.c
213 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/fcc.h
214 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/scc.h
215 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ser_cc.c
216 usr/src/lib/gss_mechs/mech_krb5/krb5/error_tables/adm_err.h
217 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/file/ktfile.h
218 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt_file.c
219 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt_srvtab.c
220 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt-int.h
221 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktadd.c
222 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktbase.c
223 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktdefault.c
224 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktfns.c
225 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktfr_entry.c
226 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktremove.c
227 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/read_servi.c
228 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_comp.c
229 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_order.c
230 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_srch.c
231 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/appdefault.c
232 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/bld_pr_ext.c
233 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/bld Princ.c
234 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/chk_trans.c
235 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/cleanup.h
236 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/conv Princ.c
237 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_addr.c
238 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_creds.c
239 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_data.c
240 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_tick.c
241 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/cp_key_ont.c
242 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/decode_kdc.c
243 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/decrypt_tk.c
244 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/deltat.c
245 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/enc_helper.c
246 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/encode_kdc.c
247 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/encrypt_tk.c
248 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/free_rtk.c
249 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/fwd_tgt.c
250 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gc_frm_kdc.c
251 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gc_via_tkt.c
252 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gen_sek.c
253 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gen_subkey.c
254 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/get_creds.c

```

4

```

255 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/get_in_tkt.c
256 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gic_keytab.c
257 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gic_opt.c
258 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gic_pwd.c
259 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/init_keyblock.c
260 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/int-proto.h
261 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/kdc_rep_dc.c
262 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/kerrs.c
263 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_error.c
264 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_priv.c
265 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_rep.c
266 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_req_ext.c
267 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_req.c
268 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_safe.c
269 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/pac.c
270 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/pr_to_salt.c
271 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/preauth.c
272 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/preauth2.c
273 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/princ_comp.c
274 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_cred.c
275 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_error.c
276 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_priv.c
277 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_rep.c
278 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_req_dec.c
279 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_req.c
280 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_safe.c
281 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/recvault.c
282 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/send_tgs.c
283 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/sendauth.c
284 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/set_realm.c
285 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/srv_reache.c
286 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/str_conv.c
287 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/tgtname.c
288 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/valid_times.c
289 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/vic_opt.c
290 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/walk_rtree.c
291 usr/src/lib/gss_mechs/mech_krb5/krb5/os/accessor.c
292 usr/src/lib/gss_mechs/mech_krb5/krb5/os/an_to_in.c
293 usr/src/lib/gss_mechs/mech_krb5/krb5/os/ccdefname.c
294 usr/src/lib/gss_mechs/mech_krb5/krb5/os/changepw.c
295 usr/src/lib/gss_mechs/mech_krb5/krb5/os/def_realm.c
296 usr/src/lib/gss_mechs/mech_krb5/krb5/os/dnsglue.c
297 usr/src/lib/gss_mechs/mech_krb5/krb5/os/dnsglue.h
298 usr/src/lib/gss_mechs/mech_krb5/krb5/os/dnssrv.c
299 usr/src/lib/gss_mechs/mech_krb5/krb5/os/foreachaddr.c
300 usr/src/lib/gss_mechs/mech_krb5/krb5/os/free_hstrl.c
301 usr/src/lib/gss_mechs/mech_krb5/krb5/os/free_krbhs.c
302 usr/src/lib/gss_mechs/mech_krb5/krb5/os/full_ipadr.c
303 usr/src/lib/gss_mechs/mech_krb5/krb5/os/gen_port.c
304 usr/src/lib/gss_mechs/mech_krb5/krb5/os/gen_rname.c
305 usr/src/lib/gss_mechs/mech_krb5/krb5/os/genaddrs.c
306 usr/src/lib/gss_mechs/mech_krb5/krb5/os/get_krbhst.c
307 usr/src/lib/gss_mechs/mech_krb5/krb5/os/gmt_mktime.c
308 usr/src/lib/gss_mechs/mech_krb5/krb5/os/hostaddr.c
309 usr/src/lib/gss_mechs/mech_krb5/krb5/os/hst_realm.c
310 usr/src/lib/gss_mechs/mech_krb5/krb5/os/ktdefname.c
311 usr/src/lib/gss_mechs/mech_krb5/krb5/os/kuserok.c
312 usr/src/lib/gss_mechs/mech_krb5/krb5/os/localaddr.c
313 usr/src/lib/gss_mechs/mech_krb5/krb5/os/locate_kdc.c
314 usr/src/lib/gss_mechs/mech_krb5/krb5/os/lock_file.c
315 usr/src/lib/gss_mechs/mech_krb5/krb5/os/mk_faddr.c
316 usr/src/lib/gss_mechs/mech_krb5/krb5/os/net_read.c
317 usr/src/lib/gss_mechs/mech_krb5/krb5/os/net_wrote.c
318 usr/src/lib/gss_mechs/mech_krb5/krb5/os/os-proto.h
319 usr/src/lib/gss_mechs/mech_krb5/krb5/os/osconfig.c
320 usr/src/lib/gss_mechs/mech_krb5/krb5/os/port2ip.c

```

```

321 usr/src/lib/gss_mechs/mech_krb5/krb5/os/prompiter.c
322 usr/src/lib/gss_mechs/mech_krb5/krb5/os/promptusr.c
323 usr/src/lib/gss_mechs/mech_krb5/krb5/os/read_msg.c
324 usr/src/lib/gss_mechs/mech_krb5/krb5/os/read_pwd.c
325 usr/src/lib/gss_mechs/mech_krb5/krb5/os/realm_dom.c
326 usr/src/lib/gss_mechs/mech_krb5/krb5/os/realm_iter.c
327 usr/src/lib/gss_mechs/mech_krb5/krb5/os/sendto_kdc.c
328 usr/src/lib/gss_mechs/mech_krb5/krb5/os/sn2princ.c
329 usr/src/lib/gss_mechs/mech_krb5/krb5/os/thread_safe.c
330 usr/src/lib/gss_mechs/mech_krb5/krb5/os/unlck_file.c
331 usr/src/lib/gss_mechs/mech_krb5/krb5/os/ustime.c
332 usr/src/lib/gss_mechs/mech_krb5/krb5/os/write_msg.c
333 usr/src/lib/gss_mechs/mech_krb5/krb5/posix/daemon.c
334 usr/src/lib/gss_mechs/mech_krb5/krb5/posix/setenv.c
335 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_base.h
336 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_conv.c
337 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_io.h
338 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_none.c
339 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc-int.h
340 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rcdef.c
341 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rcfns.c
342 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/ser_rc.c
343 usr/src/lib/gss_mechs/mech_krb5/mech/accept_sec_context.c
344 usr/src/lib/gss_mechs/mech_krb5/mech/acquire_cred_with_pw.c
345 usr/src/lib/gss_mechs/mech_krb5/mech/acquire_cred.c
346 usr/src/lib/gss_mechs/mech_krb5/mech/add_cred.c
347 usr/src/lib/gss_mechs/mech_krb5/mech/compare_name.c
348 usr/src/lib/gss_mechs/mech_krb5/mech/context_time.c
349 usr/src/lib/gss_mechs/mech_krb5/mech/copy_ccache.c
350 usr/src/lib/gss_mechs/mech_krb5/mech/disp_com_err_status.c
351 usr/src/lib/gss_mechs/mech_krb5/mech/disp_major_status.c
352 usr/src/lib/gss_mechs/mech_krb5/mech/disp_name.c
353 usr/src/lib/gss_mechs/mech_krb5/mech/disp_status.c
354 usr/src/lib/gss_mechs/mech_krb5/mech/export_name.c
355 usr/src/lib/gss_mechs/mech_krb5/mech/export_sec_context.c
356 usr/src/lib/gss_mechs/mech_krb5/mech/get_tkt_flags.c
357 usr/src/lib/gss_mechs/mech_krb5/mech/gss_libint.h
358 usr/src/lib/gss_mechs/mech_krb5/mech/import_name.c
359 usr/src/lib/gss_mechs/mech_krb5/mech/indicate_mechs.c
360 usr/src/lib/gss_mechs/mech_krb5/mech/init_sec_context.c
361 usr/src/lib/gss_mechs/mech_krb5/mech/inq_context.c
362 usr/src/lib/gss_mechs/mech_krb5/mech/inq_cred.c
363 usr/src/lib/gss_mechs/mech_krb5/mech/inq_names.c
364 usr/src/lib/gss_mechs/mech_krb5/mech/krb5_gss_glue.c
365 usr/src/lib/gss_mechs/mech_krb5/mech/lucid_context.c
366 usr/src/lib/gss_mechs/mech_krb5/mech/oid_ops.c
367 usr/src/lib/gss_mechs/mech_krb5/mech/process_context_token.c
368 usr/src/lib/gss_mechs/mech_krb5/mech/rel_buffer.c
369 usr/src/lib/gss_mechs/mech_krb5/mech/rel_cred.c
370 usr/src/lib/gss_mechs/mech_krb5/mech/rel_name.c
371 usr/src/lib/gss_mechs/mech_krb5/mech/rel_oid_set.c
372 usr/src/lib/gss_mechs/mech_krb5/mech/rel_oid.c
373 usr/src/lib/gss_mechs/mech_krb5/mech/set_allowable_ectypes.c
374 usr/src/lib/gss_mechs/mech_krb5/mech/set_ccache.c
375 usr/src/lib/gss_mechs/mech_krb5/mech/util_buffer_set.c
376 usr/src/lib/gss_mechs/mech_krb5/mech/util_buffer.c
377 usr/src/lib/gss_mechs/mech_krb5/mech/util_cksum.c
378 usr/src/lib/gss_mechs/mech_krb5/mech/util_ctxsetup.c
379 usr/src/lib/gss_mechs/mech_krb5/mech/util_dup.c
380 usr/src/lib/gss_mechs/mech_krb5/mech/util_localhost.c
381 usr/src/lib/gss_mechs/mech_krb5/mech/utl_nohash_validate.c
382 usr/src/lib/gss_mechs/mech_krb5/profile/prof_err.h
383 usr/src/lib/gss_mechs/mech_krb5/profile/prof_get.c
384 usr/src/lib/gss_mechs/mech_krb5/profile/prof_set.c
385 usr/src/lib/gss_mechs/mech_krb5/support/errors.c
386 usr/src/lib/gss_mechs/mech_krb5/support/fake-addrinfo.c

```

new/exception_lists/cstyle

```

387 usr/src/lib/gss_mechs/mech_krb5/support/init-addrinfo.c
388 usr/src/lib/gss_mechs/mech_krb5/support/plugins.c
389 usr/src/lib/gss_mechs/mech_krb5/support/supp-int.h
390 usr/src/lib/gss_mechs/mech_krb5/support/threads.c
391 usr/src/lib/gss_mechs/mech_krb5/support/utf8_conv.c
392 usr/src/lib/gss_mechs/mech_krb5/support/utf8.c
393 usr/src/lib/krb5/dyn/dyn_append.c
394 usr/src/lib/krb5/dyn/dyn_create.c
395 usr/src/lib/krb5/dyn/dyn_debug.c
396 usr/src/lib/krb5/dyn/dyn_delete.c
397 usr/src/lib/krb5/dyn/dyn_initzero.c
398 usr/src/lib/krb5/dyn/dyn_insert.c
399 usr/src/lib/krb5/dyn/dyn_paranoid.c
400 usr/src/lib/krb5/dyn/dyn_put.c
401 usr/src/lib/krb5/dyn/dyn_realloc.c
402 usr/src/lib/krb5/dyn/dyn_size.c
403 usr/src/lib/krb5/kadm5/admin_internal.h
404 usr/src/lib/krb5/kadm5/admin_xdr.h
405 usr/src/lib/krb5/kadm5/admin.h
406 usr/src/lib/krb5/kadm5/alt_prof.c
407 usr/src/lib/krb5/kadm5/chpass_util_strings.h
408 usr/src/lib/krb5/kadm5/chpass_util.c
409 usr/src/lib/krb5/kadm5/clnt/changepw.c
410 usr/src/lib/krb5/kadm5/clnt/client_handle.c
411 usr/src/lib/krb5/kadm5/clnt/client_init.c
412 usr/src/lib/krb5/kadm5/clnt/client_internal.h
413 usr/src/lib/krb5/kadm5/clnt/client_principal.c
414 usr/src/lib/krb5/kadm5/clnt/client_rpc.c
415 usr/src/lib/krb5/kadm5/clnt/clnt_chpass_util.c
416 usr/src/lib/krb5/kadm5/clnt/clnt_policy.c
417 usr/src/lib/krb5/kadm5/clnt/clnt_privs.c
418 usr/src/lib/krb5/kadm5/clnt/logger.c
419 usr/src/lib/krb5/kadm5/kadm_err.h
420 usr/src/lib/krb5/kadm5/kadm_rpc_xdr.c
421 usr/src/lib/krb5/kadm5/kadm_rpc.h
422 usr/src/lib/krb5/kadm5/misc_free.c
423 usr/src/lib/krb5/kadm5/server_internal.h
424 usr/src/lib/krb5/kadm5/srv/adb_xdr.c
425 usr/src/lib/krb5/kadm5/srv/chgpasswd.c
426 usr/src/lib/krb5/kadm5/srv/logger.c
427 usr/src/lib/krb5/kadm5/srv/server_acl.c
428 usr/src/lib/krb5/kadm5/srv/server_acl.h
429 usr/src/lib/krb5/kadm5/srv/server_dict.c
430 usr/src/lib/krb5/kadm5/srv/server_handle.c
431 usr/src/lib/krb5/kadm5/srv/server_init.c
432 usr/src/lib/krb5/kadm5/srv/server_kdb.c
433 usr/src/lib/krb5/kadm5/srv/server_misc.c
434 usr/src/lib/krb5/kadm5/srv/svr_chpass_util.c
435 usr/src/lib/krb5/kadm5/srv/svr_iters.c
436 usr/src/lib/krb5/kadm5/srv/svr_misc_free.c
437 usr/src/lib/krb5/kadm5/srv/svr_policy.c
438 usr/src/lib/krb5/kadm5/srv/svr_principal.c
439 usr/src/lib/krb5/kadm5/srv/xdr_alloc.c
440 usr/src/lib/krb5/kadm5/str_conv.c
441 usr/src/lib/krb5/kdb/adb_err.h
442 usr/src/lib/krb5/kdb/decrypt_key.c
443 usr/src/lib/krb5/kdb/encrypt_key.c
444 usr/src/lib/krb5/kdb/kdb_cpw.c
445 usr/src/lib/krb5/kdb/kdb_default.c
446 usr/src/lib/krb5/kdb/kdb5.c
447 usr/src/lib/krb5/kdb/kdb5.h
448 usr/src/lib/krb5/kdb/keytab.c
449 usr/src/lib/krb5/plugins/kdb/db2/adb_openclose.c
450 usr/src/lib/krb5/plugins/kdb/db2/adb_policy.c
451 usr/src/lib/krb5/plugins/kdb/db2/db2_exp.c
452 usr/src/lib/krb5/plugins/kdb/db2/kdb_compat.h

```

7

new/exception_lists/cstyle

```

453 usr/src/lib/krb5/plugins/kdb/db2/kdb_db2.c
454 usr/src/lib/krb5/plugins/kdb/db2/kdb_db2.h
455 usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.c
456 usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.h
457 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_close.c
458 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_conv.c
459 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_debug.c
460 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_delete.c
461 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_get.c
462 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_open.c
463 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_overflow.c
464 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_page.c
465 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_put.c
466 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_search.c
467 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_seq.c
468 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_split.c
469 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_utils.c
470 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/btree.h
471 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/extern.h
472 usr/src/lib/krb5/plugins/kdb/db2/libdb2/db/db.c
473 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/dbm.c
474 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/extern.h
475 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_bigkey.c
476 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_func.c
477 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_log2.c
478 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_page.c
479 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash.c
480 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash.h
481 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hsearch.c
482 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/page.h
483 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/search.h
484 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-int.h
485 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-ndbm.h
486 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-queue.h
487 usr/src/lib/krb5/plugins/kdb/db2/libdb2/mpool/mpool.c
488 usr/src/lib/krb5/plugins/kdb/db2/libdb2/mpool/mpool.h
489 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/extern.h
490 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_close.c
491 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_delete.c
492 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_get.c
493 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_open.c
494 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_put.c
495 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_search.c
496 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_seq.c
497 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_utils.c
498 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/recno.h
499 usr/src/lib/krb5/plugins/kdb/db2/pol_xdr.c
500 usr/src/lib/krb5/plugins/kdb/db2/policy_db.h
501 usr/src/lib/krb5/plugins/kdb/ldap/ldap_exp.c
502 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_ldap_conn.c
503 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_ldap.c
504 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_ldap.h
505 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.c
506 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.h
507 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_create.c
508 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.c
509 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.h
510 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_fetch_mkey.c
511 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.c
512 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.h
513 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.c
514 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.h
515 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_main.h
516 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_misc.c
517 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_misc.h
518 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal.c

```

8

```

519 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal.h
520 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal2.c
521 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_pwd_policy.c
522 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_pwd_policy.h
523 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_realm.c
524 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_realm.h
525 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_rights.c
526 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_stash.c
527 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_stash.h
528 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.c
529 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.h
530 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.c
531 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.h
532 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.c
533 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.h
534 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_clnt.c
535 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto_openssl.c
536 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto_openssl.h
537 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto.h
538 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_identity.c
539 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_lib.c
540 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_matching.c
541 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_profile.c
542 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_srv.c
543 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit.h
544 usr/src/lib/krb5/ss/copyright.h
545 usr/src/lib/krb5/ss/data.c
546 usr/src/lib/krb5/ss/error.c
547 usr/src/lib/krb5/ss/execute_cmd.c
548 usr/src/lib/krb5/ss/help.c
549 usr/src/lib/krb5/ss/invocation.c
550 usr/src/lib/krb5/ss/list_rqs.c
551 usr/src/lib/krb5/ss/listen.c
552 usr/src/lib/krb5/ss/mit-sipb-copyright.h
553 usr/src/lib/krb5/ss/mk_cmds.c
554 usr/src/lib/krb5/ss/options.c
555 usr/src/lib/krb5/ss/pager.c
556 usr/src/lib/krb5/ss/parse.c
557 usr/src/lib/krb5/ss/prompt.c
558 usr/src/lib/krb5/ss/request_tbl.c
559 usr/src/lib/krb5/ss/requests.c
560 usr/src/lib/krb5/ss/ss_internal.h
561 usr/src/lib/krb5/ss/ss.h
562 usr/src/lib/krb5/ss/std_rqs.c
563 usr/src/lib/krb5/ss/utills.c
564 usr/src/lib/libgss/g_glue.c
565 usr/src/lib/librtp/common/base.h
566 usr/src/lib/librtp/common/choose.h
567 usr/src/lib/librtp/common/edge.c
568 usr/src/lib/librtp/common/edge.h
569 usr/src/lib/librtp/common/migrate.c
570 usr/src/lib/librtp/common/migrate.h
571 usr/src/lib/librtp/common/p2p.c
572 usr/src/lib/librtp/common/p2p.h
573 usr/src/lib/librtp/common/pcost.c
574 usr/src/lib/librtp/common/pcost.h
575 usr/src/lib/librtp/common/port.c
576 usr/src/lib/librtp/common/port.h
577 usr/src/lib/librtp/common/portinfo.c
578 usr/src/lib/librtp/common/portinfo.h
579 usr/src/lib/librtp/common/rolesel.c
580 usr/src/lib/librtp/common/rolesel.h
581 usr/src/lib/librtp/common/roletrns.c
582 usr/src/lib/librtp/common/roletrns.h
583 usr/src/lib/librtp/common/statmch.c
584 usr/src/lib/librtp/common/statmch.h

```

```

585 usr/src/lib/librtp/common/stp_bpdu.h
586 usr/src/lib/librtp/common/stp_in.c
587 usr/src/lib/librtp/common/stp_in.h
588 usr/src/lib/librtp/common/stp_to.h
589 usr/src/lib/librtp/common/stp_vectors.h
590 usr/src/lib/librtp/common/stpm.c
591 usr/src/lib/librtp/common/stpm.h
592 usr/src/lib/librtp/common/stpmgmt.c
593 usr/src/lib/librtp/common/sttrans.c
594 usr/src/lib/librtp/common/sttrans.h
595 usr/src/lib/librtp/common/times.c
596 usr/src/lib/librtp/common/times.h
597 usr/src/lib/librtp/common/topoch.c
598 usr/src/lib/librtp/common/topoch.h
599 usr/src/lib/librtp/common/transmit.c
600 usr/src/lib/librtp/common/transmit.h
601 usr/src/lib/librtp/common/uid_stp.h
602 usr/src/lib/librtp/common/vector.c
603 usr/src/lib/librtp/common/vector.h
604 usr/src/uts/common/gssapi/gssapi.h
605 usr/src/uts/common/gssapi/mechs/krb5/crypto/block_size.c
606 usr/src/uts/common/gssapi/mechs/krb5/crypto/checksum_length.c
607 usr/src/uts/common/gssapi/mechs/krb5/crypto/cksumentypes.c
608 usr/src/uts/common/gssapi/mechs/krb5/crypto/combine_keys.c
609 usr/src/uts/common/gssapi/mechs/krb5/crypto/crc32/crc32.c
610 usr/src/uts/common/gssapi/mechs/krb5/crypto/decrypt.c
611 usr/src/uts/common/gssapi/mechs/krb5/crypto/default_state.c
612 usr/src/uts/common/gssapi/mechs/krb5/crypto/des/d3_cbc.c
613 usr/src/uts/common/gssapi/mechs/krb5/crypto/des/f_cbc.c
614 usr/src/uts/common/gssapi/mechs/krb5/crypto/des/f_parity.c
615 usr/src/uts/common/gssapi/mechs/krb5/crypto/des/weak_key.c
616 usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/checksum.c
617 usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/derive.c
618 usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/dk_decrypt.c
619 usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/dk_encrypt.c
620 usr/src/uts/common/gssapi/mechs/krb5/crypto/enc_provider/arcfour_provider.c
621 usr/src/uts/common/gssapi/mechs/krb5/crypto/enc_provider/des.c
622 usr/src/uts/common/gssapi/mechs/krb5/crypto/enc_provider/des3.c
623 usr/src/uts/common/gssapi/mechs/krb5/crypto/encrypt_length.c
624 usr/src/uts/common/gssapi/mechs/krb5/crypto/encrypt.c
625 usr/src/uts/common/gssapi/mechs/krb5/crypto/etypes.c
626 usr/src/uts/common/gssapi/mechs/krb5/crypto/hash_provider/hash_crc32.c
627 usr/src/uts/common/gssapi/mechs/krb5/crypto/hash_provider/hash_kmd5.c
628 usr/src/uts/common/gssapi/mechs/krb5/crypto/hash_provider/hash_ksha1.c
629 usr/src/uts/common/gssapi/mechs/krb5/crypto/hmac.c
630 usr/src/uts/common/gssapi/mechs/krb5/crypto/keyhash_provider/descbc.c
631 usr/src/uts/common/gssapi/mechs/krb5/crypto/keyhash_provider/k_hmac_md5.c
632 usr/src/uts/common/gssapi/mechs/krb5/crypto/keyhash_provider/k5_kmd5des.c
633 usr/src/uts/common/gssapi/mechs/krb5/crypto/make_checksum.c
634 usr/src/uts/common/gssapi/mechs/krb5/crypto/mandatory_sumtype.c
635 usr/src/uts/common/gssapi/mechs/krb5/crypto/nfold.c
636 usr/src/uts/common/gssapi/mechs/krb5/crypto/old/old_decrypt.c
637 usr/src/uts/common/gssapi/mechs/krb5/crypto/old/old_encrypt.c
638 usr/src/uts/common/gssapi/mechs/krb5/crypto/prng.c
639 usr/src/uts/common/gssapi/mechs/krb5/crypto/raw/raw_decrypt.c
640 usr/src/uts/common/gssapi/mechs/krb5/crypto/raw/raw_encrypt.c
641 usr/src/uts/common/gssapi/mechs/krb5/crypto/verify_checksum.c
642 usr/src/uts/common/gssapi/mechs/krb5/include/aes_s2k.h
643 usr/src/uts/common/gssapi/mechs/krb5/include/auth_con.h
644 usr/src/uts/common/gssapi/mechs/krb5/include/cksumentypes.h
645 usr/src/uts/common/gssapi/mechs/krb5/include/crc-32.h
646 usr/src/uts/common/gssapi/mechs/krb5/include/des_int.h
647 usr/src/uts/common/gssapi/mechs/krb5/include/dk.h
648 usr/src/uts/common/gssapi/mechs/krb5/include/enc_provider.h
649 usr/src/uts/common/gssapi/mechs/krb5/include/etypes.h
650 usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_generic.h

```

```

651 usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_krb5.h
652 usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_generic.h
653 usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_krb5.h
654 usr/src/uts/common/gssapi/mechs/krb5/include/hash_provider.h
655 usr/src/uts/common/gssapi/mechs/krb5/include/k5-int.h
656 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_16.h
657 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_32.h
658 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_64.h
659 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_16.h
660 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_32.h
661 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_64.h
662 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform.h
663 usr/src/uts/common/gssapi/mechs/krb5/include/k5-thread.h
664 usr/src/uts/common/gssapi/mechs/krb5/include/keyhash_provider.h
665 usr/src/uts/common/gssapi/mechs/krb5/include/krb5.h
666 usr/src/uts/common/gssapi/mechs/krb5/include/old.h
667 usr/src/uts/common/gssapi/mechs/krb5/include/raw.h
668 usr/src/uts/common/gssapi/mechs/krb5/include/rsa-md4.h
669 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_athctr.c
670 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_auth.c
671 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_cksum.c
672 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_key.c
673 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_princ.c
674 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/init_ctx.c
675 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/kfree.c
676 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/parse.c
677 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_actx.c
678 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_adata.c
679 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_addr.c
680 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_auth.c
681 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_cksum.c
682 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_ctx.c
683 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_key.c
684 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_princ.c
685 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/serialize.c
686 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/unparse.c
687 usr/src/uts/common/gssapi/mechs/krb5/krb5/os/c_ustime.c
688 usr/src/uts/common/gssapi/mechs/krb5/krb5/os/init_os_ctx.c
689 usr/src/uts/common/gssapi/mechs/krb5/krb5/os/timeofday.c
690 usr/src/uts/common/gssapi/mechs/krb5/krb5/os/toffset.c
691 usr/src/uts/common/gssapi/mechs/krb5/mech/delete_sec_context.c
692 usr/src/uts/common/gssapi/mechs/krb5/mech/gssapi_krb5.c
693 usr/src/uts/common/gssapi/mechs/krb5/mech/import_sec_context.c
694 usr/src/uts/common/gssapi/mechs/krb5/mech/k5seal.c
695 usr/src/uts/common/gssapi/mechs/krb5/mech/k5sealv3.c
696 usr/src/uts/common/gssapi/mechs/krb5/mech/k5unseal.c
697 usr/src/uts/common/gssapi/mechs/krb5/mech/seal.c
698 usr/src/uts/common/gssapi/mechs/krb5/mech/ser_sctx.c
699 usr/src/uts/common/gssapi/mechs/krb5/mech/sign.c
700 usr/src/uts/common/gssapi/mechs/krb5/mech/unseal.c
701 usr/src/uts/common/gssapi/mechs/krb5/mech/util_crypt.c
702 usr/src/uts/common/gssapi/mechs/krb5/mech/util_ordering.c
703 usr/src/uts/common/gssapi/mechs/krb5/mech/util_seed.c
704 usr/src/uts/common/gssapi/mechs/krb5/mech/util_seqnum.c
705 usr/src/uts/common/gssapi/mechs/krb5/mech/util_set.c
706 usr/src/uts/common/gssapi/mechs/krb5/mech/util_token.c
707 usr/src/uts/common/gssapi/mechs/krb5/mech/util_validate.c
708 usr/src/uts/common/gssapi/mechs/krb5/mech/val_cred.c
709 usr/src/uts/common/gssapi/mechs/krb5/mech/verify.c
710 usr/src/uts/common/gssapi/mechs/krb5/mech/wrap_size_limit.c
711 usr/src/uts/common/io/e1000api/e1000_80003es2lan.c
712 usr/src/uts/common/io/e1000api/e1000_80003es2lan.h
713 usr/src/uts/common/io/e1000api/e1000_82540.c
714 usr/src/uts/common/io/e1000api/e1000_82541.c
715 usr/src/uts/common/io/e1000api/e1000_82541.h
716 usr/src/uts/common/io/e1000api/e1000_82542.c

```

```

717 usr/src/uts/common/io/e1000api/e1000_82543.c
718 usr/src/uts/common/io/e1000api/e1000_82543.h
719 usr/src/uts/common/io/e1000api/e1000_82571.c
720 usr/src/uts/common/io/e1000api/e1000_82571.h
721 usr/src/uts/common/io/e1000api/e1000_82575.c
722 usr/src/uts/common/io/e1000api/e1000_82575.h
723 usr/src/uts/common/io/e1000api/e1000_api.c
724 usr/src/uts/common/io/e1000api/e1000_api.h
725 usr/src/uts/common/io/e1000api/e1000_defines.h
726 usr/src/uts/common/io/e1000api/e1000_hw.h
727 usr/src/uts/common/io/e1000api/e1000_i210.c
728 usr/src/uts/common/io/e1000api/e1000_i210.h
729 usr/src/uts/common/io/e1000api/e1000_ich8lan.c
730 usr/src/uts/common/io/e1000api/e1000_ich8lan.h
731 usr/src/uts/common/io/e1000api/e1000_mac.c
732 usr/src/uts/common/io/e1000api/e1000_mac.h
733 usr/src/uts/common/io/e1000api/e1000_manage.c
734 usr/src/uts/common/io/e1000api/e1000_manage.h
735 usr/src/uts/common/io/e1000api/e1000_mbx.c
736 usr/src/uts/common/io/e1000api/e1000_mbx.h
737 usr/src/uts/common/io/e1000api/e1000_nvmm.c
738 usr/src/uts/common/io/e1000api/e1000_nvmm.h
739 usr/src/uts/common/io/e1000api/e1000_phy.c
740 usr/src/uts/common/io/e1000api/e1000_phy.h
741 usr/src/uts/common/io/e1000api/e1000_regs.h
742 usr/src/uts/common/io/e1000api/e1000_vf.c
743 usr/src/uts/common/io/e1000api/e1000_vf.h
744 usr/src/uts/common/io/ixgbe/ixgbe_82598.c
745 usr/src/uts/common/io/ixgbe/ixgbe_82598.h
746 usr/src/uts/common/io/ixgbe/ixgbe_82599.c
747 usr/src/uts/common/io/ixgbe/ixgbe_82599.h
748 usr/src/uts/common/io/ixgbe/ixgbe_api.c
749 usr/src/uts/common/io/ixgbe/ixgbe_api.h
750 usr/src/uts/common/io/ixgbe/ixgbe_common.c
751 usr/src/uts/common/io/ixgbe/ixgbe_common.h
752 usr/src/uts/common/io/ixgbe/ixgbe_mbx.c
753 usr/src/uts/common/io/ixgbe/ixgbe_mbx.h
754 usr/src/uts/common/io/ixgbe/ixgbe_osdep.h
755 usr/src/uts/common/io/ixgbe/ixgbe_phy.c
756 usr/src/uts/common/io/ixgbe/ixgbe_phy.h
757 usr/src/uts/common/io/ixgbe/ixgbe_type.h
758 usr/src/uts/common/io/ixgbe/ixgbe_x540.c
759 usr/src/uts/common/io/ixgbe/ixgbe_x540.h
760 usr/src/uts/intel/io/acpica/debugger/dbcmds.c
761 usr/src/uts/intel/io/acpica/debugger/dbdisply.c
762 usr/src/uts/intel/io/acpica/debugger/dbexec.c
763 usr/src/uts/intel/io/acpica/debugger/dbfileio.c
764 usr/src/uts/intel/io/acpica/debugger/dbhistory.c
765 usr/src/uts/intel/io/acpica/debugger/dbinput.c
766 usr/src/uts/intel/io/acpica/debugger/dbmethod.c
767 usr/src/uts/intel/io/acpica/debugger/dbnames.c
768 usr/src/uts/intel/io/acpica/debugger/dbstats.c
769 usr/src/uts/intel/io/acpica/debugger/dbutils.c
770 usr/src/uts/intel/io/acpica/debugger/dbxface.c
771 usr/src/uts/intel/io/acpica/disassembler/dmbuffer.c
772 usr/src/uts/intel/io/acpica/disassembler/dmnames.c
773 usr/src/uts/intel/io/acpica/disassembler/dmobjects.c
774 usr/src/uts/intel/io/acpica/disassembler/dmopcode.c
775 usr/src/uts/intel/io/acpica/disassembler/dmresrc.c
776 usr/src/uts/intel/io/acpica/disassembler/dmresrc1.c
777 usr/src/uts/intel/io/acpica/disassembler/dmresrcs.c
778 usr/src/uts/intel/io/acpica/disassembler/dmutils.c
779 usr/src/uts/intel/io/acpica/disassembler/dmwalk.c
780 usr/src/uts/intel/io/acpica/dispatcher/dsargs.c
781 usr/src/uts/intel/io/acpica/dispatcher/dscontrol.c
782 usr/src/uts/intel/io/acpica/dispatcher/dsfield.c

```

```

783 usr/src/uts/intel/io/acpica/dispatcher/dsinit.c
784 usr/src/uts/intel/io/acpica/dispatcher/dsmethod.c
785 usr/src/uts/intel/io/acpica/dispatcher/dsmthdat.c
786 usr/src/uts/intel/io/acpica/dispatcher/dsobject.c
787 usr/src/uts/intel/io/acpica/dispatcher/dsopcode.c
788 usr/src/uts/intel/io/acpica/dispatcher/dsutils.c
789 usr/src/uts/intel/io/acpica/dispatcher/dswexec.c
790 usr/src/uts/intel/io/acpica/dispatcher/dswload.c
791 usr/src/uts/intel/io/acpica/dispatcher/dswload2.c
792 usr/src/uts/intel/io/acpica/dispatcher/dswscope.c
793 usr/src/uts/intel/io/acpica/dispatcher/dswstate.c
794 usr/src/uts/intel/io/acpica/events/evevent.c
795 usr/src/uts/intel/io/acpica/events/evglock.c
796 usr/src/uts/intel/io/acpica/events/evgpe.c
797 usr/src/uts/intel/io/acpica/events/evgpeblk.c
798 usr/src/uts/intel/io/acpica/events/evgpeinit.c
799 usr/src/uts/intel/io/acpica/events/evgpeutil.c
800 usr/src/uts/intel/io/acpica/events/evmisc.c
801 usr/src/uts/intel/io/acpica/events/evregion.c
802 usr/src/uts/intel/io/acpica/events/evrgnini.c
803 usr/src/uts/intel/io/acpica/events/evsci.c
804 usr/src/uts/intel/io/acpica/events/evxface.c
805 usr/src/uts/intel/io/acpica/events/evxfevnt.c
806 usr/src/uts/intel/io/acpica/events/evxfppe.c
807 usr/src/uts/intel/io/acpica/events/evxfreg.c
808 usr/src/uts/intel/io/acpica/executor/exconfig.c
809 usr/src/uts/intel/io/acpica/executor/exconvrt.c
810 usr/src/uts/intel/io/acpica/executor/excreate.c
811 usr/src/uts/intel/io/acpica/executor/exdebug.c
812 usr/src/uts/intel/io/acpica/executor/exdump.c
813 usr/src/uts/intel/io/acpica/executor/exfield.c
814 usr/src/uts/intel/io/acpica/executor/exfldio.c
815 usr/src/uts/intel/io/acpica/executor/exmisc.c
816 usr/src/uts/intel/io/acpica/executor/exmutex.c
817 usr/src/uts/intel/io/acpica/executor/exnames.c
818 usr/src/uts/intel/io/acpica/executor/exoparg1.c
819 usr/src/uts/intel/io/acpica/executor/exoparg2.c
820 usr/src/uts/intel/io/acpica/executor/exoparg3.c
821 usr/src/uts/intel/io/acpica/executor/exoparg6.c
822 usr/src/uts/intel/io/acpica/executor/exprep.c
823 usr/src/uts/intel/io/acpica/executor/exregion.c
824 usr/src/uts/intel/io/acpica/executor/exresnte.c
825 usr/src/uts/intel/io/acpica/executor/exresolv.c
826 usr/src/uts/intel/io/acpica/executor/exresop.c
827 usr/src/uts/intel/io/acpica/executor/exstore.c
828 usr/src/uts/intel/io/acpica/executor/exstoren.c
829 usr/src/uts/intel/io/acpica/executor/exstorob.c
830 usr/src/uts/intel/io/acpica/executor/exsystem.c
831 usr/src/uts/intel/io/acpica/executor/exutils.c
832 usr/src/uts/intel/io/acpica/hardware/hwacpi.c
833 usr/src/uts/intel/io/acpica/hardware/hwpci.c
834 usr/src/uts/intel/io/acpica/hardware/hwpci.c
835 usr/src/uts/intel/io/acpica/hardware/hwregs.c
836 usr/src/uts/intel/io/acpica/hardware/hwsleep.c
837 usr/src/uts/intel/io/acpica/hardware/hwtimer.c
838 usr/src/uts/intel/io/acpica/hardware/hwvalid.c
839 usr/src/uts/intel/io/acpica/hardware/hwxface.c
840 usr/src/uts/intel/io/acpica/namespace/nsaccess.c
841 usr/src/uts/intel/io/acpica/namespace/nsalloc.c
842 usr/src/uts/intel/io/acpica/namespace/nsdump.c
843 usr/src/uts/intel/io/acpica/namespace/nsdumpdv.c
844 usr/src/uts/intel/io/acpica/namespace/nseval.c
845 usr/src/uts/intel/io/acpica/namespace/nsinit.c
846 usr/src/uts/intel/io/acpica/namespace/nsload.c
847 usr/src/uts/intel/io/acpica/namespace/nsnames.c
848 usr/src/uts/intel/io/acpica/namespace/nsobject.c

```

```

849 usr/src/uts/intel/io/acpica/namespace/nsparse.c
850 usr/src/uts/intel/io/acpica/namespace/nspredef.c
851 usr/src/uts/intel/io/acpica/namespace/nsrepair.c
852 usr/src/uts/intel/io/acpica/namespace/nsrepair2.c
853 usr/src/uts/intel/io/acpica/namespace/nssearch.c
854 usr/src/uts/intel/io/acpica/namespace/nsutils.c
855 usr/src/uts/intel/io/acpica/namespace/nswalk.c
856 usr/src/uts/intel/io/acpica/namespace/nsxfeval.c
857 usr/src/uts/intel/io/acpica/namespace/nsxfname.c
858 usr/src/uts/intel/io/acpica/namespace/nsxfobj.c
859 usr/src/uts/intel/io/acpica/parser/psargs.c
860 usr/src/uts/intel/io/acpica/parser/psloop.c
861 usr/src/uts/intel/io/acpica/parser/psopcode.c
862 usr/src/uts/intel/io/acpica/parser/psparse.c
863 usr/src/uts/intel/io/acpica/parser/psscope.c
864 usr/src/uts/intel/io/acpica/parser/psree.c
865 usr/src/uts/intel/io/acpica/parser/psutils.c
866 usr/src/uts/intel/io/acpica/parser/pswalk.c
867 usr/src/uts/intel/io/acpica/parser/psxface.c
868 usr/src/uts/intel/io/acpica/resources/rsaddr.c
869 usr/src/uts/intel/io/acpica/resources/rsalc.c
870 usr/src/uts/intel/io/acpica/resources/rscreate.c
871 usr/src/uts/intel/io/acpica/resources/rsdump.c
872 usr/src/uts/intel/io/acpica/resources/rsinfo.c
873 usr/src/uts/intel/io/acpica/resources/rsio.c
874 usr/src/uts/intel/io/acpica/resources/rsirq.c
875 usr/src/uts/intel/io/acpica/resources/rslist.c
876 usr/src/uts/intel/io/acpica/resources/rsmemory.c
877 usr/src/uts/intel/io/acpica/resources/rsmisc.c
878 usr/src/uts/intel/io/acpica/resources/rsutils.c
879 usr/src/uts/intel/io/acpica/resources/rsxface.c
880 usr/src/uts/intel/io/acpica/tables/tbfadt.c
881 usr/src/uts/intel/io/acpica/tables/tbfind.c
882 usr/src/uts/intel/io/acpica/tables/tbinstal.c
883 usr/src/uts/intel/io/acpica/tables/tbutils.c
884 usr/src/uts/intel/io/acpica/tables/tbxface.c
885 usr/src/uts/intel/io/acpica/tables/tbxfront.c
886 usr/src/uts/intel/io/acpica/utilities/utalloc.c
887 usr/src/uts/intel/io/acpica/utilities/utcache.c
888 usr/src/uts/intel/io/acpica/utilities/utclib.c
889 usr/src/uts/intel/io/acpica/utilities/utcopy.c
890 usr/src/uts/intel/io/acpica/utilities/utdebug.c
891 usr/src/uts/intel/io/acpica/utilities/utdecode.c
892 usr/src/uts/intel/io/acpica/utilities/utdelete.c
893 usr/src/uts/intel/io/acpica/utilities/uteval.c
894 usr/src/uts/intel/io/acpica/utilities/utglobal.c
895 usr/src/uts/intel/io/acpica/utilities/utids.c
896 usr/src/uts/intel/io/acpica/utilities/utinit.c
897 usr/src/uts/intel/io/acpica/utilities/utlock.c
898 usr/src/uts/intel/io/acpica/utilities/utmath.c
899 usr/src/uts/intel/io/acpica/utilities/utmisc.c
900 usr/src/uts/intel/io/acpica/utilities/utmutex.c
901 usr/src/uts/intel/io/acpica/utilities/utobject.c
902 usr/src/uts/intel/io/acpica/utilities/utosi.c
903 usr/src/uts/intel/io/acpica/utilities/utresrc.c
904 usr/src/uts/intel/io/acpica/utilities/utstate.c
905 usr/src/uts/intel/io/acpica/utilities/uttrack.c
906 usr/src/uts/intel/io/acpica/utilities/utxface.c
907 usr/src/uts/intel/io/acpica/utilities/utxferror.c
908 usr/src/uts/intel/sys/acpi/acapps.h
909 usr/src/uts/intel/sys/acpi/accommon.h
910 usr/src/uts/intel/sys/acpi/acconfig.h
911 usr/src/uts/intel/sys/acpi/acdebug.h
912 usr/src/uts/intel/sys/acpi/acdisasm.h
913 usr/src/uts/intel/sys/acpi/acdispat.h
914 usr/src/uts/intel/sys/acpi/acevents.h

```

915 usr/src/uts/intel/sys/acpi/acexcep.h
916 usr/src/uts/intel/sys/acpi/acglobal.h
917 usr/src/uts/intel/sys/acpi/achware.h
918 usr/src/uts/intel/sys/acpi/acinterp.h
919 usr/src/uts/intel/sys/acpi/aclocal.h
920 usr/src/uts/intel/sys/acpi/acmacros.h
921 usr/src/uts/intel/sys/acpi/acnames.h
922 usr/src/uts/intel/sys/acpi/acnamesp.h
923 usr/src/uts/intel/sys/acpi/acobject.h
924 usr/src/uts/intel/sys/acpi/acopcode.h
925 usr/src/uts/intel/sys/acpi/acoutput.h
926 usr/src/uts/intel/sys/acpi/acparser.h
927 usr/src/uts/intel/sys/acpi/acpi.h
928 usr/src/uts/intel/sys/acpi/acpiosxf.h
929 usr/src/uts/intel/sys/acpi/acpixf.h
930 usr/src/uts/intel/sys/acpi/acpredef.h
931 usr/src/uts/intel/sys/acpi/acresrc.h
932 usr/src/uts/intel/sys/acpi/acrestyp.h
933 usr/src/uts/intel/sys/acpi/acstruct.h
934 usr/src/uts/intel/sys/acpi/actables.h
935 usr/src/uts/intel/sys/acpi/actbl.h
936 usr/src/uts/intel/sys/acpi/actbl1.h
937 usr/src/uts/intel/sys/acpi/actbl2.h
938 usr/src/uts/intel/sys/acpi/actypes.h
939 usr/src/uts/intel/sys/acpi/acutils.h
940 usr/src/uts/intel/sys/acpi/amlcode.h
941 usr/src/uts/intel/sys/acpi/amlresrc.h
942 usr/src/uts/intel/sys/acpi/platform/accygwin.h
943 usr/src/uts/intel/sys/acpi/platform/acefi.h
944 usr/src/uts/intel/sys/acpi/platform/acenv.h
945 usr/src/uts/intel/sys/acpi/platform/acfreebsd.h
946 usr/src/uts/intel/sys/acpi/platform/acgcc.h
947 usr/src/uts/intel/sys/acpi/platform/acintel.h
948 usr/src/uts/intel/sys/acpi/platform/aclinux.h
949 usr/src/uts/intel/sys/acpi/platform/acmsvc.h
950 usr/src/uts/intel/sys/acpi/platform/acnetbsd.h
951 usr/src/uts/intel/sys/acpi/platform/acos2.h
952 usr/src/uts/intel/sys/acpi/platform/acsolaris.h
953 usr/src/uts/intel/sys/acpi/platform/acwin.h
954 usr/src/uts/intel/sys/acpi/platform/acwin64.h

new/exception_lists/hdrchk

1

9208 Sat Jul 19 19:24:27 2014

new/exception_lists/hdrchk

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

- 1 usr/src/cmd/krb5/kadmin/cli/kadmin.h
- 2 usr/src/cmd/krb5/kadmin/dbutil/import_err.h
- 3 usr/src/cmd/krb5/kadmin/dbutil/kdb5_util.h
- 4 usr/src/cmd/krb5/kadmin/dbutil/nstrtok.h
- 5 usr/src/cmd/krb5/kadmin/dbutil/string_table.h
- 6 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd_strings.h
- 7 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd.h
- 8 usr/src/cmd/krb5/kadmin/ktutil/ktutil.h
- 9 usr/src/cmd/krb5/kadmin/server/misc.h
- 10 usr/src/cmd/krb5/krb5kdc/extern.h
- 11 usr/src/cmd/krb5/krb5kdc/kdc_util.h
- 12 usr/src/cmd/krb5/krb5kdc/policy.h
- 13 usr/src/cmd/krb5/ldap_util/kdb5_ldap_list.h
- 14 usr/src/cmd/krb5/ldap_util/kdb5_ldap_policy.h
- 15 usr/src/cmd/krb5/ldap_util/kdb5_ldap_realm.h
- 16 usr/src/cmd/krb5/ldap_util/kdb5_ldap_services.h
- 17 usr/src/cmd/krb5/ldap_util/kdb5_ldap_util.h
- 18 usr/src/cmd/krb5/slave/kprop.h
- 19 usr/src/cmd/localedef/localedef.h
- 20 **usr/src/cmd/mandoc/config.h**
- 21 **usr/src/cmd/mandoc/html.h**
- 22 **usr/src/cmd/mandoc/libman.h**
- 23 **usr/src/cmd/mandoc/libmandoc.h**
- 24 **usr/src/cmd/mandoc/libmdoc.h**
- 25 **usr/src/cmd/mandoc/libroff.h**
- 26 **usr/src/cmd/mandoc/main.h**
- 27 **usr/src/cmd/mandoc/man.h**
- 28 **usr/src/cmd/mandoc/mandoc.h**
- 29 **usr/src/cmd/mandoc/mdoc.h**
- 30 **usr/src/cmd/mandoc/out.h**
- 31 **usr/src/cmd/mandoc/term.h**
- 32 usr/src/common/openssl/crypto/krb5/krb5_asn.h
- 33 usr/src/lib/gss_mechs/mech_krb5/et/error_table.h
- 34 usr/src/lib/gss_mechs/mech_krb5/et/internal.h
- 35 usr/src/lib/gss_mechs/mech_krb5/et/mit-sipb-copyright.h
- 36 usr/src/lib/gss_mechs/mech_krb5/include/cache-addrinfo.h
- 37 usr/src/lib/gss_mechs/mech_krb5/include/cm.h
- 38 usr/src/lib/gss_mechs/mech_krb5/include/com_err.h
- 39 usr/src/lib/gss_mechs/mech_krb5/include/db-config.h
- 40 usr/src/lib/gss_mechs/mech_krb5/include/db.h
- 41 usr/src/lib/gss_mechs/mech_krb5/include/fake-addrinfo.h
- 42 usr/src/lib/gss_mechs/mech_krb5/include/foreachaddr.h
- 43 usr/src/lib/gss_mechs/mech_krb5/include/k5-int-pkinit.h
- 44 usr/src/lib/gss_mechs/mech_krb5/include/k5-utf8.h
- 45 usr/src/lib/gss_mechs/mech_krb5/include/kdb_kt.h
- 46 usr/src/lib/gss_mechs/mech_krb5/include/krb5_libinit.h
- 47 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_defs.h
- 48 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_proto.h
- 49 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm.h
- 50 usr/src/lib/gss_mechs/mech_krb5/include/krb5/copyright.h
- 51 usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-err.h
- 52 usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-plugin.h
- 53 usr/src/lib/gss_mechs/mech_krb5/include/krb5/kdb_dbc.h
- 54 usr/src/lib/gss_mechs/mech_krb5/include/krb5/kdb.h
- 55 usr/src/lib/gss_mechs/mech_krb5/include/locate_plugin.h
- 56 usr/src/lib/gss_mechs/mech_krb5/include/osconf.h

new/exception_lists/hdrchk

2

- 57 usr/src/lib/gss_mechs/mech_krb5/include/port-sockets.h
- 58 usr/src/lib/gss_mechs/mech_krb5/include/preauth_plugin.h
- 59 usr/src/lib/gss_mechs/mech_krb5/include/socket-utils.h
- 60 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.h
- 61 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.h
- 62 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.h
- 63 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_decode.h
- 64 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_encode.h
- 65 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.h
- 66 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.h
- 67 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1buf.h
- 68 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krbasn1.h
- 69 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc-int.h
- 70 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/fcc.h
- 71 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/scc.h
- 72 usr/src/lib/gss_mechs/mech_krb5/krb5/error_tables/adm_err.h
- 73 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/file/ktfile.h
- 74 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt-int.h
- 75 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/cleanup.h
- 76 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/int-proto.h
- 77 usr/src/lib/gss_mechs/mech_krb5/krb5/os/dnsglue.h
- 78 usr/src/lib/gss_mechs/mech_krb5/krb5/os/os-proto.h
- 79 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_base.h
- 80 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_io.h
- 81 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc-int.h
- 82 usr/src/lib/gss_mechs/mech_krb5/mech/gss_libinit.h
- 83 usr/src/lib/gss_mechs/mech_krb5/profile/prof_err.h
- 84 usr/src/lib/gss_mechs/mech_krb5/support/supp-int.h
- 85 usr/src/lib/krb5/kadm5/admin_internal.h
- 86 usr/src/lib/krb5/kadm5/admin_xdr.h
- 87 usr/src/lib/krb5/kadm5/admin.h
- 88 usr/src/lib/krb5/kadm5/chpass_util_strings.h
- 89 usr/src/lib/krb5/kadm5/clnt/client_internal.h
- 90 usr/src/lib/krb5/kadm5/kadm_err.h
- 91 usr/src/lib/krb5/kadm5/kadm_rpc.h
- 92 usr/src/lib/krb5/kadm5/server_internal.h
- 93 usr/src/lib/krb5/kadm5/srv/server_acl.h
- 94 usr/src/lib/krb5/kdb/adb_err.h
- 95 usr/src/lib/krb5/kdb/kdb5.h
- 96 usr/src/lib/krb5/plugins/kdb/db2/kdb_compat.h
- 97 usr/src/lib/krb5/plugins/kdb/db2/kdb_db2.h
- 98 usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.h
- 99 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/btree.h
- 100 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/extern.h
- 101 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/extern.h
- 102 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash.h
- 103 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/page.h
- 104 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/search.h
- 105 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-int.h
- 106 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-ndbm.h
- 107 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-queue.h
- 108 usr/src/lib/krb5/plugins/kdb/db2/libdb2/mpool/mpool.h
- 109 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/extern.h
- 110 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/recno.h
- 111 usr/src/lib/krb5/plugins/kdb/db2/policy_db.h
- 112 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_ldap.h
- 113 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.h
- 114 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.h
- 115 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.h
- 116 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.h
- 117 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_main.h
- 118 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_misc.h
- 119 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal.h
- 120 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_pwd_policy.h
- 121 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_realm.h
- 122 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_stash.h

new/exception_lists/hdrchk

3

123 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.h
124 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.h
125 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.h
126 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto_openssl.h
127 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto.h
128 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit.h
129 usr/src/lib/krb5/ss/copyright.h
130 usr/src/lib/krb5/ss/mit-sipb-copyright.h
131 usr/src/lib/krb5/ss/ss_internal.h
132 usr/src/lib/krb5/ss/ss.h
133 usr/src/lib/libc/port/locale/utills.h
134 usr/src/lib/librstp/common/base.h
135 usr/src/lib/librstp/common/choose.h
136 usr/src/lib/librstp/common/edge.h
137 usr/src/lib/librstp/common/migrate.h
138 usr/src/lib/librstp/common/p2p.h
139 usr/src/lib/librstp/common/pcost.h
140 usr/src/lib/librstp/common/port.h
141 usr/src/lib/librstp/common/portinfo.h
142 usr/src/lib/librstp/common/rolesel.h
143 usr/src/lib/librstp/common/roletrns.h
144 usr/src/lib/librstp/common/statmch.h
145 usr/src/lib/librstp/common/stp_bpdu.h
146 usr/src/lib/librstp/common/stp_in.h
147 usr/src/lib/librstp/common/stp_to.h
148 usr/src/lib/librstp/common/stp_vectors.h
149 usr/src/lib/librstp/common/stpm.h
150 usr/src/lib/librstp/common/sttrans.h
151 usr/src/lib/librstp/common/times.h
152 usr/src/lib/librstp/common/topoch.h
153 usr/src/lib/librstp/common/transmit.h
154 usr/src/lib/librstp/common/uid_stp.h
155 usr/src/lib/librstp/common/vector.h
156 usr/src/uts/common/gssapi/mechs/krb5/include/aes_s2k.h
157 usr/src/uts/common/gssapi/mechs/krb5/include/auth_con.h
158 usr/src/uts/common/gssapi/mechs/krb5/include/cksumtypes.h
159 usr/src/uts/common/gssapi/mechs/krb5/include/crc-32.h
160 usr/src/uts/common/gssapi/mechs/krb5/include/des_int.h
161 usr/src/uts/common/gssapi/mechs/krb5/include/dk.h
162 usr/src/uts/common/gssapi/mechs/krb5/include/enc_provider.h
163 usr/src/uts/common/gssapi/mechs/krb5/include/etypes.h
164 usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_generic.h
165 usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_krb5.h
166 usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_generic.h
167 usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_krb5.h
168 usr/src/uts/common/gssapi/mechs/krb5/include/hash_provider.h
169 usr/src/uts/common/gssapi/mechs/krb5/include/k5-int.h
170 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_16.h
171 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_32.h
172 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_64.h
173 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_16.h
174 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_32.h
175 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_64.h
176 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform.h
177 usr/src/uts/common/gssapi/mechs/krb5/include/k5-thread.h
178 usr/src/uts/common/gssapi/mechs/krb5/include/keyhash_provider.h
179 usr/src/uts/common/gssapi/mechs/krb5/include/krb5.h
180 usr/src/uts/common/gssapi/mechs/krb5/include/old.h
181 usr/src/uts/common/gssapi/mechs/krb5/include/raw.h
182 usr/src/uts/common/gssapi/mechs/krb5/include/rsa-md4.h
183 usr/src/uts/common/io/ixgbe/ixgbe_common.h
184 usr/src/uts/intel/sys/acpi/acdebug.h
185 usr/src/uts/intel/sys/acpi/acdisasm.h
186 usr/src/uts/intel/sys/acpi/acevents.h
187 usr/src/uts/intel/sys/acpi/acinterp.h
188 usr/src/uts/intel/sys/acpi/acmacros.h

new/exception_lists/hdrchk

4

189 usr/src/uts/intel/sys/acpi/acnames.h
190 usr/src/uts/intel/sys/acpi/acpredef.h
191 usr/src/uts/intel/sys/acpi/acresrc.h
192 usr/src/uts/intel/sys/acpi/acstruct.h
193 usr/src/uts/intel/sys/acpi/amlresrc.h
194 usr/src/uts/intel/sys/acpi/platform/acwin64.h

new/exception_lists/packaging

1

```

*****
27930 Sat Jul 19 19:24:27 2014
new/exception_lists/packaging
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright 2011 Nexenta Systems, Inc. All rights reserved.
25 # Copyright 2012 OmniTI Computer Consulting, Inc. All rights reserved.
26 #
27 #
28 #
29 # Exception List for validate_pkg
30 #
31 #
32 #
33 # The following entries are built in the /proto area
34 # but not included in any packages - this is intentional.
35 #
36 usr/include/auth_list.h
37 usr/include/bsm/audit_door_infc.h
38 usr/include/bsm/audit_private.h
39 usr/include/bsm/devalloc.h
40 usr/include/getxby_door.h
41 usr/include/passwdutil.h
42 usr/include/priv_utils.h
43 usr/include/rpcsvc/daemon_utils.h
44 usr/include/rpcsvc/svc_dg_priv.h
45 usr/include/security/pam_impl.h
46 usr/include/sys/clock_impl.h
47 usr/include/sys/ieeefp.h
48 usr/include/sys/winlockio.h
49 usr/include/scsi/plugins/ses/vendor/sun_impl.h
50 #
51 # Private/Internal libraries of the Cryptographic Framework.
52 #
53 lib/libkcf.so
54 lib/llib-lelfsign
55 lib/llib-lelfsign.ln
56 lib/llib-lkcf

```

new/exception_lists/packaging

2

```

57 lib/llib-lkcf.ln
58 usr/include/libelfsign.h
59 usr/lib/llib-lsoftcrypto
60 usr/lib/llib-lsoftcrypto.ln
61 usr/lib/amd64/llib-lsoftcrypto.ln      i386
62 usr/lib/sparcv9/llib-lsoftcrypto.ln   sparc
63 #
64 #
65 # The following files are used by the DHCP service, the
66 # standalone's DHCP implementation, and the kernel (nfs_dlboot).
67 # They contain interfaces which are currently private.
68 #
69 usr/include/dhcp_svc_confkey.h
70 usr/include/dhcp_svc_confopt.h
71 usr/include/dhcp_svc_private.h
72 usr/include/dhcp_symbol.h
73 usr/include/sys/sunos_dhcp_class.h
74 usr/lib/libdhcpsvc.so
75 usr/lib/llib-ldhcpsvc
76 usr/lib/llib-ldhcpsvc.ln
77 #
78 # Private MAC driver header files
79 #
80 usr/include/inet/iptun.h
81 usr/include/sys/aggr_impl.h
82 usr/include/sys/aggr.h
83 usr/include/sys/dld_impl.h
84 usr/include/sys/dld_ioc.h
85 usr/include/sys/dls_impl.h
86 usr/include/sys/dls.h
87 usr/include/sys/mac_client_impl.h
88 usr/include/sys/mac_client.h
89 usr/include/sys/mac_flow_impl.h
90 usr/include/sys/mac_impl.h
91 usr/include/sys/mac_soft_ring.h
92 usr/include/sys/mac_stat.h
93 #
94 # Private GLDv3 userland libraries and headers
95 #
96 usr/include/libdladm_impl.h
97 usr/include/libdlaggr.h
98 usr/include/libdlether.h
99 usr/include/libdlflow_impl.h
100 usr/include/libdlflow.h
101 usr/include/libdliptun.h
102 usr/include/libdlmgmt.h
103 usr/include/libdlsim.h
104 usr/include/libdlstat.h
105 usr/include/libdlvnic.h
106 usr/include/libdlwlan_impl.h
107 usr/include/libdlwlan.h
108 #
109 # Virtual Network Interface Card (VNIC)
110 #
111 usr/include/sys/vnic.h
112 usr/include/sys/vnic_impl.h
113 #
114 # Private libipadm lint library and header files
115 #
116 usr/include/ipadm_ipmgmt.h
117 usr/include/ipadm_ndpd.h
118 usr/include/libipadm.h
119 lib/llib-lipadm
120 lib/llib-lipadm.ln
121 lib/libipadm.so
122 #

```

new/exception_lists/packaging

3

```

123 # Private libsocket header file
124 #
125 usr/include/libsocket_priv.h
126 #
127 # IKE and IPsec support library exceptions. The IKE support
128 # library contains exclusively private interfaces, as does
129 # libipseutil. My apologies for the glut of header files here.
130 #
131 usr/include/errfp.h
132 usr/include/ikedoor.h
133 usr/include/ipsec_util.h
134 usr/lib/libike.so
135 usr/lib/amd64/libike.so          i386
136 usr/lib/sparcv9/libike.so      sparc
137 usr/lib/libipseutil.so
138 usr/lib/amd64/libipseutil.so   i386
139 usr/lib/sparcv9/libipseutil.so sparc
140 usr/lib/llib-like
141 usr/lib/llib-like.ln
142 usr/lib/amd64/llib-like.ln     i386
143 usr/lib/sparcv9/llib-like.ln  sparc
144 usr/lib/llib-lipseutil
145 usr/lib/llib-lipseutil.ln
146 usr/lib/amd64/llib-lipseutil.ln i386
147 usr/lib/sparcv9/llib-lipseutil.ln sparc
148 #
149 usr/include/inet/ip_impl.h
150 usr/include/inet/ip_ndp.h
151 usr/include/inet/ip2mac_impl.h
152 usr/include/inet/ip2mac.h
153 usr/include/inet/rawip_impl.h
154 usr/include/inet/tcp_impl.h
155 usr/include/inet/udp_impl.h
156 usr/include/libmail.h
157 usr/include/libnwam_priv.h
158 usr/include/protocols/ripngd.h
159 usr/include/s_string.h
160 usr/include/sys/logindmux_impl.h
161 usr/include/sys/vgareg.h
162 #
163 # Some IPsec headers can't be shipped lest we hit export controls...
164 #
165 usr/include/inet/ipsec_impl.h
166 usr/include/inet/ipsec_info.h
167 usr/include/inet/ipsecah.h
168 usr/include/inet/ipsecesp.h
169 usr/include/inet/keysock.h
170 usr/include/inet/sadb.h
171 usr/include/sys/shal_consts.h
172 usr/include/sys/sha2_consts.h
173 #
174 #
175 # Filtering out directories not shipped
176 #
177 usr/4lib                          i386
178 #
179 # These files contain definitions shared privately between the kernel
180 # and libc. There is no reason for them to be part of a package that
181 # a customer should ever see. They are installed in the proto area by
182 # the uts build because libc and other components, like truss, are
183 # dependent upon their contents and should not have their own copies.
184 #
185 usr/include/sys/libc_kernel.h
186 usr/include/sys/synch32.h
187 #
188 # These files are installed in the proto area by the build of libproc for

```

new/exception_lists/packaging

4

```

189 # the benefit of the builds of cmd/truss, cmd/gcore and cmd/ptools, which
190 # use libproc as their common process-control library. These are not
191 # interfaces for customer use, so the files are excluded from packaging.
192 #
193 lib/llib-lproc
194 lib/llib-lproc.ln
195 lib/amd64/llib-lproc.ln          i386
196 lib/sparcv9/llib-lproc.ln      sparc
197 usr/include/libproc.h
198 #
199 # Private interfaces for libdisasm
200 #
201 usr/include/libdisasm.h
202 usr/lib/llib-ldisasm
203 usr/lib/llib-ldisasm.ln
204 usr/lib/amd64/llib-ldisasm.ln   i386
205 usr/lib/sparcv9/llib-ldisasm.ln sparc
206 #
207 # Private interfaces for libraidcfg
208 #
209 usr/include/raidcfg_spi.h
210 usr/include/raidcfg.h
211 usr/lib/libraidcfg.so
212 usr/lib/amd64/libraidcfg.so      i386
213 usr/lib/sparcv9/libraidcfg.so   sparc
214 usr/lib/llib-lraidcfg
215 usr/lib/llib-lraidcfg.ln
216 usr/lib/amd64/llib-lraidcfg.ln  i386
217 usr/lib/sparcv9/llib-lraidcfg.ln sparc
218 #
219 # This file is used for private communication between mdb, drv/kmdb, and
220 # misc/kmdb. The interfaces described herein are not intended for customer
221 # use, and are thus excluded from packaging.
222 #
223 usr/include/sys/kmdb.h
224 #
225 # These files are installed in the proto area by the build of libdhcpage
226 # and libdhcputil for the benefit of DHCP-related networking commands such
227 # as dhcpage, dhcpageinfo, ifconfig, and netstat. These are not interfaces
228 # for customer use, so the files are excluded from packaging.
229 #
230 lib/libdhcpage.so
231 lib/libdhcputil.so
232 lib/amd64/libdhcputil.so          i386
233 lib/sparcv9/libdhcputil.so      sparc
234 lib/llib-ldhcpage
235 lib/llib-ldhcpage.ln
236 lib/llib-ldhcputil
237 lib/llib-ldhcputil.ln
238 lib/amd64/llib-ldhcputil.ln     i386
239 lib/sparcv9/llib-ldhcputil.ln   sparc
240 usr/include/dhcp_hostconf.h
241 usr/include/dhcp_impl.h
242 usr/include/dhcp_inittab.h
243 usr/include/dhcp_stable.h
244 usr/include/dhcp_symbol_common.h
245 usr/include/dhcpage_ipc.h
246 usr/include/dhcpage_util.h
247 usr/include/dhcpage_msg.h
248 usr/lib/libdhcpage.so
249 usr/lib/libdhcputil.so
250 usr/lib/amd64/libdhcputil.so     i386
251 usr/lib/sparcv9/libdhcputil.so   sparc
252 usr/lib/llib-ldhcpage
253 usr/lib/llib-ldhcpage.ln
254 usr/lib/llib-ldhcputil

```

new/exception_lists/packaging

5

```

255 usr/lib/llib-ldhcputil.ln
256 usr/lib/amd64/llib-ldhcputil.ln      i386
257 usr/lib/sparcv9/llib-ldhcputil.ln    sparc
258 #
259 # These files are installed in the proto area by the build of libinstzones
260 # and libpkg
261 #
262 usr/lib/llib-linstzones
263 usr/lib/llib-linstzones.ln
264 usr/lib/amd64/llib-linstzones.ln      i386
265 usr/lib/sparcv9/llib-linstzones.ln    sparc
266 usr/lib/llib-lpkg
267 usr/lib/llib-lpkg.ln
268 #
269 # Don't ship header files private to libipmp and in.mpathd
270 #
271 usr/include/ipmp_query_impl.h
272 #
273 # These files are installed in the proto area by the build of libinetsvc,
274 # an inetd-specific library shared by inetd, inetadm and inetconv. Only
275 # the shared object is shipped.
276 #
277 usr/include/inetsvc.h
278 usr/lib/libinetsvc.so
279 usr/lib/llib-linetsvc
280 usr/lib/llib-linetsvc.ln
281 #
282 # These files are installed in the proto area by the build of libinetutil,
283 # a general purpose library for the benefit of internet utilities. Only
284 # the shared object is shipped.
285 #
286 lib/libinetutil.so
287 lib/amd64/libinetutil.so      i386
288 lib/sparcv9/libinetutil.so    sparc
289 lib/llib-linetutil
290 lib/llib-linetutil.ln
291 lib/amd64/llib-linetutil.ln  i386
292 lib/sparcv9/llib-linetutil.ln sparc
293 usr/include/libinetutil.h
294 usr/include/netinet/inetutil.h
295 usr/include/ofmt.h
296 usr/lib/libinetutil.so
297 usr/lib/amd64/libinetutil.so  i386
298 usr/lib/sparcv9/libinetutil.so sparc
299 usr/lib/llib-linetutil
300 usr/lib/llib-linetutil.ln
301 usr/lib/amd64/llib-linetutil.ln i386
302 usr/lib/sparcv9/llib-linetutil.ln sparc
303 #
304 # Miscellaneous kernel interfaces or kernel->user interfaces that are
305 # consolidation private and we do not want to export at this time.
306 #
307 usr/include/sys/cryptmod.h
308 usr/include/sys/dumpadm.h
309 usr/include/sys/ontrap.h
310 usr/include/sys/sysmsg_impl.h
311 usr/include/sys/vlan.h
312 #
313 # These files are installed in the proto area so lvm can use
314 # them during the build process.
315 #
316 lib/llib-lmeta
317 lib/llib-lmeta.ln
318 usr/include/sdssc.h
319 usr/lib/llib-lmeta
320 usr/lib/llib-lmeta.ln

```

new/exception_lists/packaging

6

```

321 #
322 # non-public pci header
323 #
324 usr/include/sys/pci_impl.h
325 usr/include/sys/pci_tools.h
326 #
327 # Exception list for RCM project, included by librcm and rcm_daemon
328 #
329 usr/include/librcm_event.h
330 usr/include/librcm_impl.h
331 #
332 # MDB deliverables that are not yet public
333 #
334 usr/lib/mdb/proc/mdb_test.so
335 usr/lib/mdb/proc/sparcv9/mdb_test.so  sparc
336 #
337 # SNCA project exception list
338 #
339 usr/include/inet/kssl/kssl.h
340 usr/include/inet/kssl/ksslimpl.h
341 usr/include/inet/kssl/ksslproto.h
342 usr/include/inet/nca
343 #
344 # these are "removed" from the source product build because the only
345 # packages that currently deliver them are removed.
346 # they really shouldn't be in here.
347 #
348 etc/sfw
349 #
350 # Entries for the libmech_krb5 symlink, which has been included
351 # for build purposes only, not delivered to customers.
352 #
353 usr/lib/gss/libmech_krb5.so
354 usr/lib/amd64/gss/libmech_krb5.so      i386
355 usr/lib/sparcv9/gss/libmech_krb5.so    sparc
356 usr/lib/libmech_krb5.so
357 usr/lib/amd64/libmech_krb5.so         i386
358 usr/lib/sparcv9/libmech_krb5.so       sparc
359 #
360 # Entries for headers from efcodes project which user does not need to see
361 #
362 usr/platform/sun4u/include/sys/fc_plat.h      sparc
363 usr/platform/sun4u/include/sys/fcode.h        sparc
364 #
365 # Private net80211 headers
366 #
367 usr/include/sys/net80211_crypto.h
368 usr/include/sys/net80211_ht.h
369 usr/include/sys/net80211_proto.h
370 usr/include/sys/net80211.h
371 #
372 usr/include/net/wpa.h
373 #
374 # PPPoE files not delivered to customers.
375 #
376 usr/include/net/pppoe.h
377 usr/include/net/sppptun.h
378 #
379 # Simnet
380 #
381 usr/include/net/simnet.h
382 #
383 # Bridging internal data structures
384 #
385 usr/include/net/bridge_impl.h
386 #

```

new/exception_lists/packaging

7

```

387 # User<->kernel interface used by cfgadm/USB only
388 #
389 usr/include/sys/usb/hubd/hubd_impl.h
390 #
391 # User<->kernel interface used by cfgadm/SATA only
392 #
393 usr/include/sys/sata/sata_cfgadm.h          i386
394 #
395 # Private ucred kernel header
396 #
397 usr/include/sys/ucred.h
398 #
399 # Private and/or platform-specific smf(5) files
400 #
401 lib/librestart.so
402 lib/llib-lrestart
403 lib/llib-lrestart.ln
404 lib/amd64/llib-lrestart.ln          i386
405 lib/sparcv9/llib-lrestart.ln       sparc
406 usr/include/libcontract_priv.h
407 usr/include/librestart_priv.h
408 usr/include/librestart.h
409 usr/lib/librestart.so
410 usr/lib/sparcv9/librestart.so       sparc
411 lib/svc/manifest/platform/sun4u    i386
412 lib/svc/manifest/platform/sun4v    i386
413 var/svc/manifest/platform/sun4u    i386
414 var/svc/manifest/platform/sun4v    i386
415 etc/svc/profile/platform_sun4v.xml i386
416 etc/svc/profile/platform_SUNW,SPARC-Enterprise.xml i386
417 etc/svc/profile/platform_SUNW,Sun-Fire-15000.xml i386
418 etc/svc/profile/platform_SUNW,Sun-Fire-880.xml i386
419 etc/svc/profile/platform_SUNW,Sun-Fire-V890.xml i386
420 etc/svc/profile/platform_SUNW,Sun-Fire.xml i386
421 etc/svc/profile/platform_SUNW,Ultra-Enterprise-10000.xml i386
422 etc/svc/profile/platform_SUNW,UltraSPARC-IIe-NetraCT-40.xml i386
423 etc/svc/profile/platform_SUNW,UltraSPARC-IIe-NetraCT-60.xml i386
424 etc/svc/profile/platform_SUNW,UltraSPARC-III-NetraCT.xml i386
425 #
426 # Private libuutil files
427 #
428 lib/libuutil.so
429 lib/llib-luutil
430 lib/llib-luutil.ln
431 lib/sparcv9/llib-luutil.ln       sparc
432 usr/include/libuutil_impl.h
433 usr/lib/libuutil.so
434 usr/lib/sparcv9/libuutil.so       sparc
435 #
436 # Private Multidata file.
437 #
438 usr/include/sys/multidata_impl.h
439 #
440 # The following files are used by wanboot.
441 # They contain interfaces which are currently private.
442 #
443 usr/include/sys/wanboot_impl.h
444 usr/include/wanboot
445 usr/include/wanbootutil.h
446 #
447 # Even though all the objects built under usr/src/stand are later glommed
448 # together into a couple of second-stage boot loaders, we dump the static
449 # archives and lint libraries into $(ROOT)/stand for intermediate use
450 # (e.g., for lint, linking the second-stage boot loaders, ...). Since
451 # these are merely intermediate objects, they do not need to be packaged.
452 #

```

new/exception_lists/packaging

8

```

453 stand                                sparc
454 #
455 # Private KCF header files
456 #
457 usr/include/sys/crypto/elfsign.h
458 usr/include/sys/crypto/impl.h
459 usr/include/sys/crypto/ops_impl.h
460 usr/include/sys/crypto/sched_impl.h
461 #
462 # The following files are installed in the proto area
463 # by the build of libavl (AVL Tree Interface Library).
464 # libavl contains interfaces which are all private interfaces.
465 #
466 lib/libavl.so
467 lib/amd64/libavl.so          i386
468 lib/sparcv9/libavl.so       sparc
469 lib/llib-lavl
470 lib/llib-lavl.ln
471 lib/amd64/llib-lavl.ln     i386
472 lib/sparcv9/llib-lavl.ln   sparc
473 usr/lib/libavl.so
474 usr/lib/amd64/libavl.so     i386
475 usr/lib/sparcv9/libavl.so   sparc
476 usr/lib/llib-lavl
477 usr/lib/llib-lavl.ln
478 usr/lib/amd64/llib-lavl.ln i386
479 usr/lib/sparcv9/llib-lavl.ln sparc
480 #
481 # The following files are installed in the proto area
482 # by the build of libcmdutils (Command Utilities Library).
483 # libcmdutils contains interfaces which are all private interfaces.
484 #
485 lib/libcmdutils.so
486 lib/amd64/libcmdutils.so     i386
487 lib/sparcv9/libcmdutils.so   sparc
488 lib/llib-lcmdutils
489 lib/llib-lcmdutils.ln
490 lib/amd64/llib-lcmdutils.ln i386
491 lib/sparcv9/llib-lcmdutils.ln sparc
492 usr/include/libcmdutils.h
493 usr/lib/libcmdutils.so
494 usr/lib/amd64/libcmdutils.so i386
495 usr/lib/sparcv9/libcmdutils.so sparc
496 usr/lib/llib-lcmdutils
497 usr/lib/llib-lcmdutils.ln
498 usr/lib/amd64/llib-lcmdutils.ln i386
499 usr/lib/sparcv9/llib-lcmdutils.ln sparc
500 #
501 # Private interfaces in libsec
502 #
503 usr/include/aclutils.h
504 #
505 # USB skeleton driver stays in sync with the rest of USB but doesn't ship.
506 #
507 kernel/drv/usbskel          i386
508 kernel/drv/amd64/usbskel    i386
509 kernel/drv/sparcv9/usbskel  sparc
510 kernel/drv/usbskel.conf
511 #
512 # Consolidation and Sun private libdevid interfaces
513 # Public libdevid interfaces provided by devid.h
514 #
515 usr/include/sys/libdevid.h
516 #
517 # The following files are installed in the proto area by the build of
518 # libprtdiag. libprtdiag contains interfaces which are all private.

```

new/exception_lists/packaging

9

```

519 # Only the shared object is shipped.
520 #
521 usr/platform/sun4u/lib/llib-lprtdiag          sparc
522 usr/platform/sun4u/lib/llib-lprtdiag.ln      sparc
523 usr/platform/sun4v/lib/llib-lprtdiag.ln      sparc
524 #
525 # The following files are installed in the proto area by the build of
526 # mdesc driver in sun4v. These header files are used on in the build
527 # and do not need to be shipped to customers.
528 #
529 usr/include/sys/mdesc.h                       sparc
530 usr/include/sys/mdesc_impl.h                 sparc
531 usr/platform/sun4v/include/sys/mach_descrip.h sparc
532 #
533 # The following files are installed in the proto area by the build of
534 # libpcp. libpcp contains interfaces which are all private.
535 # Only the shared object is shipped.
536 #
537 usr/platform/sun4v/lib/llib-lpcp.ln          sparc
538 usr/platform/SUNW,Netra-CP3060/lib/llib-lpcp.ln sparc
539 usr/platform/SUNW,Netra-CP3260/lib/llib-lpcp.ln sparc
540 usr/platform/SUNW,Netra-T5220/lib/llib-lpcp.ln sparc
541 usr/platform/SUNW,Netra-T5440/lib/llib-lpcp.ln sparc
542 usr/platform/SUNW,SPARC-Enterprise-T5120/lib/llib-lpcp.ln sparc
543 usr/platform/SUNW,Sun-Blade-T6300/lib/llib-lpcp.ln sparc
544 usr/platform/SUNW,Sun-Blade-T6320/lib/llib-lpcp.ln sparc
545 usr/platform/SUNW,Sun-Fire-T200/lib/llib-lpcp.ln sparc
546 usr/platform/SUNW,T5140/lib/llib-lpcp.ln     sparc
547 usr/platform/SUNW,USBRDT-5240/lib/llib-lpcp.ln sparc
548 #
549 # ZFS internal tools and lint libraries
550 #
551 usr/lib/llib-lzfs_jni
552 usr/lib/llib-lzfs_jni.ln
553 usr/lib/amd64/llib-lzfs_jni.ln               i386
554 usr/lib/sparcv9/llib-lzfs_jni.ln            sparc
555 usr/lib/llib-lzpool
556 usr/lib/llib-lzpool.ln                       i386
557 usr/lib/amd64/llib-lzpool.ln                 i386
558 usr/lib/sparcv9/llib-lzpool.ln              sparc
559 #
560 # ZFS JNI headers
561 #
562 usr/include/libzfs_jni_dataset.h
563 usr/include/libzfs_jni_disk.h
564 usr/include/libzfs_jni_diskmgt.h
565 usr/include/libzfs_jni_ipool.h
566 usr/include/libzfs_jni_main.h
567 usr/include/libzfs_jni_pool.h
568 usr/include/libzfs_jni_property.h
569 usr/include/libzfs_jni_util.h
570 #
571 # These files are installed in the proto area for Solaris scsi_vhci driver
572 # (for MPAPI support) and should not be shipped
573 #
574 usr/include/sys/scsi/adapters/mpapi_impl.h
575 usr/include/sys/scsi/adapters/mpapi_scsi_vhci.h
576 #
577 # This library is installed in the proto area by the build of libdisasm, and is
578 # only used when building the KMDB disasm module.
579 #
580 usr/lib/libstanddisasm.so
581 usr/lib/amd64/libstanddisasm.so               i386
582 usr/lib/sparcv9/libstanddisasm.so            sparc
583 #
584 # TSol: tsol doesn't ship lint source, and tsnet isn't for customers at all.

```

new/exception_lists/packaging

10

```

585 #
586 lib/libtsnet.so
587 usr/lib/llib-ltsnet
588 usr/lib/llib-ltsol
589 #
590 # nss interfaces shared between libnsl and other ON libraries.
591 #
592 usr/include/nss.h
593 #
594 # AT&T AST (ksh93) files which are currently needed only to build OS/Net
595 # (msgcc&co.)
596 # libast
597 usr/lib/libast.so
598 usr/lib/amd64/libast.so                       i386
599 usr/lib/sparcv9/libast.so                     sparc
600 usr/lib/llib-last
601 usr/lib/llib-last.ln
602 usr/lib/amd64/llib-last.ln                  i386
603 usr/lib/sparcv9/llib-last.ln                sparc
604 # libcmd
605 usr/lib/llib-lcmd
606 usr/lib/llib-lcmd.ln
607 usr/lib/amd64/llib-lcmd.ln                  i386
608 usr/lib/sparcv9/llib-lcmd.ln                sparc
609 # libdll
610 usr/lib/libdll.so
611 usr/lib/amd64/libdll.so                       i386
612 usr/lib/sparcv9/libdll.so                    sparc
613 usr/lib/llib-ldll
614 usr/lib/llib-ldll.ln
615 usr/lib/amd64/llib-ldll.ln                  i386
616 usr/lib/sparcv9/llib-ldll.ln                sparc
617 # libpp (a helper library needed by AST's msgcc)
618 usr/lib/libpp.so
619 usr/lib/llib-lpp
620 usr/lib/llib-lpp.ln
621 usr/lib/locale/C/LC_MESSAGES/libpp
622 # libshell
623 usr/lib/libshell.so
624 usr/lib/amd64/libshell.so                     i386
625 usr/lib/sparcv9/libshell.so                  sparc
626 usr/lib/llib-lshell
627 usr/lib/llib-lshell.ln
628 usr/lib/amd64/llib-lshell.ln                i386
629 usr/lib/sparcv9/llib-lshell.ln              sparc
630 # libsum
631 usr/lib/libsum.so
632 usr/lib/amd64/libsum.so                       i386
633 usr/lib/sparcv9/libsum.so                     sparc
634 usr/lib/llib-lsum
635 usr/lib/llib-lsum.ln
636 usr/lib/amd64/llib-lsum.ln                  i386
637 usr/lib/sparcv9/llib-lsum.ln                sparc
638 #
639 # This file is used in ON to build DSCP clients. It is not for customers.
640 #
641 usr/include/libdscp.h                         sparc
642 #
643 # These files are used by the iSCSI Target and the iSCSI Initiator
644 #
645 usr/include/sys/iscsi_protocol.h
646 usr/include/sys/iscsi_authclient.h
647 usr/include/sys/iscsi_authclientglue.h
648 #
649 # These files are used by the COMSTAR iSCSI target port provider
650 #

```

```

651 usr/include/sys/idm
652 usr/include/sys/iscsit/chap.h
653 usr/include/sys/iscsit/iscsi_if.h
654 usr/include/sys/iscsit/iscsi_protocol.h
655 usr/include/sys/iscsit/radius_packet.h
656 usr/include/sys/iscsit/radius_protocol.h
657 #
658 # libshare is private and the 64-bit sharemgr is not delivered.
659 #
660 usr/lib/libshare.so
661 usr/lib/amd64/libshare.so          i386
662 usr/lib/sparcv9/libshare.so       sparc
663 usr/lib/fs/autofs/libshare_autofs.so
664 usr/lib/fs/autofs/amd64/libshare_autofs.so          i386
665 usr/lib/fs/autofs/sparcv9/libshare_autofs.so       sparc
666 usr/lib/fs/nfs/libshare_nfs.so
667 usr/lib/fs/nfs/amd64/libshare_nfs.so          i386
668 usr/lib/fs/nfs/sparcv9/libshare_nfs.so         sparc
669 usr/lib/fs/smb/libshare_smb.so
670 usr/lib/fs/smb/amd64/libshare_smb.so          i386
671 usr/lib/fs/smb/sparcv9/libshare_smb.so        sparc
672 usr/lib/fs/smbfs/libshare_smbfs.so
673 usr/lib/fs/smbfs/amd64/libshare_smbfs.so       i386
674 usr/lib/fs/smbfs/sparcv9/libshare_smbfs.so     sparc
675 usr/include/libshare_impl.h
676 usr/include/scfutil.h
677 #
678 # These files are installed in the proto area by the build of libpri for
679 # the benefit of the builds of FMA libldom, Zeus, picld plugins, and/or
680 # other libpri consumers. However, the libpri interfaces are private to
681 # Sun (Consolidation Private) and not intended for customer use. So these
682 # files (the symlink and the lint library) are excluded from packaging.
683 #
684 usr/lib/libpri.so                 sparc
685 usr/lib/llib-lpri                sparc
686 usr/lib/llib-lpri.ln             sparc
687 usr/lib/sparcv9/libpri.so         sparc
688 usr/lib/sparcv9/llib-lpri.ln     sparc
689 #
690 # These files are installed in the proto area by the build of libds for
691 # the benefit of the builds of sun4v IO FMA and/or other libds
692 # consumers. However, the libds interfaces are private to Sun
693 # (Consolidation Private) and not intended for customer use. So these
694 # files (the symlink and the lint library) are excluded from packaging.
695 #
696 usr/lib/libds.so                 sparc
697 usr/lib/sparcv9/libds.so         sparc
698 usr/lib/llib-lds                sparc
699 usr/lib/llib-lds.ln             sparc
700 usr/lib/sparcv9/llib-lds.ln     sparc
701 usr/lib/libdsconfig.so
702 usr/lib/llib-ldscfg.ln
703 usr/platform/sun4v/include/sys/libds.h  sparc
704 usr/platform/sun4v/include/sys/vlds.h  sparc
705 #
706 # Private/Internal u8_textprep header file. Do not ship.
707 #
708 usr/include/sys/u8_textprep_data.h
709 #
710 # SQLite is private, used by SMF (svc.configd), idmapd and libsmf.
711 #
712 usr/include/sqlite
713 usr/lib/libsqlite-native.o
714 usr/lib/libsqlite.o
715 usr/lib/llib-lsqlite.ln
716 usr/lib/smbfs/libsqlite.so

```

```

717 #
718 # Private/Internal kiconv header files. Do not ship.
719 #
720 usr/include/sys/kiconv_big5_utf8.h
721 usr/include/sys/kiconv_ck_common.h
722 usr/include/sys/kiconv_cp950hkscs_utf8.h
723 usr/include/sys/kiconv_emeal.h
724 usr/include/sys/kiconv_emea2.h
725 usr/include/sys/kiconv_euckr_utf8.h
726 usr/include/sys/kiconv_euctw_utf8.h
727 usr/include/sys/kiconv_gb18030_utf8.h
728 usr/include/sys/kiconv_gb2312_utf8.h
729 usr/include/sys/kiconv_hkscs_utf8.h
730 usr/include/sys/kiconv_ja_jis_to_unicode.h
731 usr/include/sys/kiconv_ja_unicode_to_jis.h
732 usr/include/sys/kiconv_ja.h
733 usr/include/sys/kiconv_ko.h
734 usr/include/sys/kiconv_latin1.h
735 usr/include/sys/kiconv_sc.h
736 usr/include/sys/kiconv_tc.h
737 usr/include/sys/kiconv_uhc_utf8.h
738 usr/include/sys/kiconv_utf8_big5.h
739 usr/include/sys/kiconv_utf8_cp950hkscs.h
740 usr/include/sys/kiconv_utf8_euckr.h
741 usr/include/sys/kiconv_utf8_euctw.h
742 usr/include/sys/kiconv_utf8_gb18030.h
743 usr/include/sys/kiconv_utf8_gb2312.h
744 usr/include/sys/kiconv_utf8_hkscs.h
745 usr/include/sys/kiconv_utf8_uhc.h
746 #
747 # At this time, the ttydefs.cleanup file is only useful on sun4u systems
748 #
749 etc/flash/postdeployment/ttydefs.cleanup          i386
750 #
751 # This header file is shared only between the power commands and
752 # ppm/srn modules # and should not be in any package
753 #
754 usr/include/sys/srn.h
755 #
756 # Private/Internal header files of smbfs. Do not ship.
757 #
758 usr/include/smb
759 usr/include/smbfs
760 #
761 # Private/Internal dtrace scripts of smbfs. Do not ship.
762 #
763 usr/lib/smbfs/dtrace
764 #
765 # Private/Internal (lint) libraries of smbfs. Do not ship.
766 #
767 usr/lib/reparse/llib-lreparse_smb
768 usr/lib/reparse/llib-lreparse_smb.ln
769 usr/lib/smbfs/llib-lmlrpc
770 usr/lib/smbfs/llib-lmlrpc.ln
771 usr/lib/smbfs/llib-lmlsvc
772 usr/lib/smbfs/llib-lmlsvc.ln
773 usr/lib/smbfs/llib-lsmb
774 usr/lib/smbfs/llib-lsmb.ln
775 usr/lib/smbfs/llib-lsmbns
776 usr/lib/smbfs/llib-lsmbns.ln
777 #
778 #
779 # Private/Internal 64-bit libraries of smbfs. Do not ship.
780 #
781 usr/lib/smbfs/amd64                i386
782 usr/lib/smbfs/sparcv9              sparc

```



```

784 usr/lib/reparse/amd64/libreparse_smb.so      i386
785 usr/lib/reparse/amd64/libreparse_smb.so.1   i386
786 usr/lib/reparse/amd64/llib-lreparse_smb.ln  i386
787 usr/lib/reparse/sparcv9/libreparse_smb.so   sparc
788 usr/lib/reparse/sparcv9/libreparse_smb.so.1 sparc
789 usr/lib/reparse/sparcv9/llib-lreparse_smb.ln sparc
790 #
791 # Private dirent, extended to include flags, for use by SMB server
792 #
793 usr/include/sys/extdirent.h
794 #
795 # Private header files for vscan service
796 #
797 usr/include/libvscan.h
798 usr/include/sys/vscan.h
799 #
800 # libvscan is private
801 #
802 usr/lib/vscan/llib-lvscan
803 usr/lib/vscan/llib-lvscan.ln
804 #
805 # i86hvm is not a full platform. It is just a home for paravirtualized
806 # drivers. There is no usr/ component to this sub-platform, but the
807 # directory is created in the proto area to keep other tools happy.
808 #
809 usr/platform/i86hvm                                i386
810 #
811 # Private sdcard framework headers
812 #
813 usr/include/sys/sdcard
814 #
815 # libsmbfs is private
816 #
817 usr/include/netsmb
818 usr/lib/libsmbfs.so
819 usr/lib/amd64/libsmbfs.so      i386
820 usr/lib/sparcv9/libsmbfs.so   sparc
821 usr/lib/llib-lsmbfs
822 usr/lib/llib-lsmbfs.ln
823 usr/lib/amd64/llib-lsmbfs.ln  i386
824 usr/lib/sparcv9/llib-lsmbfs.ln sparc
825 #
826 # demo & test program for smbfs (private) ACL support
827 #
828 usr/lib/fs/smbfs/chacl
829 usr/lib/fs/smbfs/lsacl
830 usr/lib/fs/smbfs/testnp
831 #
832 # FC related files
833 kernel/kmdb/fcipc                i386
834 kernel/kmdb/amd64/fcipc          i386
835 kernel/kmdb/sparcv9/fcipc       sparc
836 kernel/kmdb/fcpc                i386
837 kernel/kmdb/amd64/fcpc          i386
838 kernel/kmdb/sparcv9/fcpc       sparc
839 kernel/kmdb/fctl                i386
840 kernel/kmdb/amd64/fctl          i386
841 kernel/kmdb/sparcv9/fctl       sparc
842 kernel/kmdb/qlc                 i386
843 kernel/kmdb/amd64/qlc          i386
844 kernel/kmdb/sparcv9/qlc       sparc
845 lib/llib-la5k                  sparc
846 lib/llib-la5k.ln               sparc
847 lib/sparcv9/llib-la5k.ln       sparc
848 lib/llib-lg_fc                  sparc

```

```

849 lib/llib-lg_fc.ln              sparc
850 lib/sparcv9/llib-lg_fc.ln      sparc
851 usr/include/a_state.h          sparc
852 usr/include/a5k.h              sparc
853 usr/include/exec.h             sparc
854 usr/include/g_scsi.h           sparc
855 usr/include/g_state.h          sparc
856 usr/include/gfc.h              sparc
857 usr/include/l_common.h         sparc
858 usr/include/l_error.h          sparc
859 usr/include/rom.h              sparc
860 usr/include/stgcom.h           sparc
861 usr/include/sys/fibre-channel
862 usr/lib/llib-LHBAAPI
863 usr/lib/llib-LHBAAPI.ln
864 usr/lib/amd64/llib-LHBAAPI.ln  i386
865 usr/lib/sparcv9/llib-LHBAAPI.ln sparc
866 #
867 usr/bin/dscfgcli
868 usr/bin/sd_diag
869 usr/bin/sd_stats
870 usr/include/nsctl.h
871 usr/include/sys/ncall
872 usr/include/sys/nsc_ddi.h
873 usr/include/sys/nsc_thread.h
874 usr/include/sys/nsctl
875 usr/include/sys/nskernd.h
876 usr/include/sys/unistat
877 usr/lib/libnsctl.so
878 usr/lib/librdc.so
879 usr/lib/libunistat.so
880 usr/lib/llib-lnsctl.ln
881 usr/lib/llib-lrdc.ln
882 usr/lib/llib-lunistat.ln
883 #
884 # These files are used by the iSCSI initiator only.
885 # No reason to ship them.
886 #
887 usr/include/sys/scsi/adapters/iscsi_door.h
888 usr/include/sys/scsi/adapters/iscsi_if.h
889 #
890 # sbd ioctl hdr
891 #
892 usr/include/sys/stmf_sbd_ioctl.h
893 #
894 # proxy port provider interface
895 #
896 usr/include/sys/pppt_ic_if.h
897 usr/include/sys/pppt_ioctl.h
898 #
899 # proxy daemon lint library
900 #
901 usr/lib/llib-lstmfproxy
902 usr/lib/llib-lstmfproxy.ln
903 usr/lib/amd64/llib-lstmfproxy.ln  i386
904 usr/lib/sparcv9/llib-lstmfproxy.ln sparc
905 #
906 # portable object file and dictionary used by libfmd_msg test
907 #
908 usr/lib/fm/dict/TEST.dict
909 usr/lib/locale/C/LC_MESSAGES/TEST.mo
910 usr/lib/locale/C/LC_MESSAGES/TEST.po
911 #
912 # Private idmap RPC protocol
913 #
914 usr/include/rpcsvc/idmap_prot.h

```

new/exception_lists/packaging

15

```

915 usr/include/rpcsvc/idmap_prot.x
916 #
917 # Private idmap directory API
918 #
919 usr/include/directory.h
920 #
921 # librstp is private for bridging
922 #
923 usr/include/stp_bpdu.h
924 usr/include/stp_in.h
925 usr/include/stp_vectors.h
926 usr/lib/librstp.so
927 usr/lib/llib-lrstp
928 usr/lib/llib-lrstp.ln
929 #
930 # Private nvfru API
931 #
932 usr/include/nvfru.h
933 #
934 # vrrp
935 #
936 usr/include/libvrrpadm.h
937 usr/lib/libvrrpadm.so
938 usr/lib/amd64/libvrrpadm.so      i386
939 usr/lib/sparcv9/libvrrpadm.so   sparc
940 usr/lib/llib-lvrrpadm
941 usr/lib/llib-lvrrpadm.ln
942 usr/lib/amd64/llib-lvrrpadm.ln  i386
943 usr/lib/sparcv9/llib-lvrrpadm.ln sparc
944 #
945 # This is only used during the -t tools build
946 #
947 opt/onbld/bin/i386/elfsign      i386
948 opt/onbld/bin/sparc/elfsign     sparc
949 opt/onbld/bin/i386/mandoc       i386
950 opt/onbld/bin/sparc/mandoc     sparc
951 #
952 #
953 # Private libdwarf
954 #
955 opt/onbld/lib/i386/libdwarf.so  i386
956 opt/onbld/lib/sparc/libdwarf.so sparc
957 #
958 #
959 # Private socket filter API
960 #
961 usr/include/sys/sockfilter.h
962 #
963 # We don't actually validate license action payloads, and the license
964 # staging area is provided as a separate basedir for package
965 # publication. The net result is that everything therein should be
966 # ignored for packaging validation.
967 #
968 licenses
969 #
970 # Libbe is private
971 #
972 usr/include/libbe_priv.h
973 #
974 # ipmi is at present only useful on i386, but for historical reasons is
975 # delivered on SPARC and used by the build.
976 #
977 usr/include/sys/ipmi.h  sparc
978 #
979 #
980 # libsaveargs is private

```

new/exception_lists/packaging

16

```

981 #
982 usr/include/saveargs.h          i386
983 usr/lib/amd64/libsaveargs.so    i386
984 usr/lib/amd64/libstandsaveargs.so i386
985 usr/lib/amd64/llib-lsaveargs.ln i386
986 #
987 #
988 # libpcidb is private
989 #
990 usr/include/pcidb.h
991 usr/lib/amd64/libpcidb.so       i386
992 usr/lib/amd64/llib-lpcidb.ln   i386
993 usr/lib/sparcv9/libpcidb.so    sparc
994 usr/lib/sparcv9/llib-lpcidb.ln sparc
995 usr/lib/libpcidb.so
996 usr/lib/llib-lpcidb
997 usr/lib/llib-lpcidb.ln

```

new/usr/src/Makefile

1

```
*****
7476 Sat Jul 19 19:24:27 2014
new/usr/src/Makefile
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright (c) 2012 by Delphix. All rights reserved.
25 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
26 #
27 #
28 #
29 # Makefile for system source
30 #
31 # include global definitions
32 # include Makefile.master
33 #
34 # the Targetdirs file is the AT&T target.dirs file in a makefile format.
35 # it defines TARGETDIRS and ROOTDIRS.
36 # include Targetdirs
37 #
38 # COMMON_SUBDIRS= uts lib cmd ucblib ucblib psm man test
39 # sparc_SUBDIRS= stand
40 # i386_SUBDIRS= grub
41 #
42 #
43 # sparc needs to build stand before psm
44 #
45 # $(SPARC_BLD)psm: stand
46 #
47 # SUBDIRS= $(COMMON_SUBDIRS) $(MACHINE)_SUBDIRS
48 #
49 # HDRSUBDIRS= uts head lib cmd
50 #
51 # UCB headers are bug-for-bug compatible and not checkable against the header
52 # standards.
53 #
54 # CHKHDRSUBDIRS= head uts lib
55 #
56 #
```

new/usr/src/Makefile

2

```
57 # Headers that can be built in parallel
58 #
59 # PARALLEL_HEADERS = sysheaders userheaders libheaders cmdheaders
60 #
61 #
62 # Directories that can be built in parallel
63 #
64 # PARALLEL_DIRS = uts lib man
65 #
66 # The check target also causes smf(5) service manifests to be validated.
67 # CHKMFSUBDIRS= cmd
68 #
69 # And man page formats
70 # CHKMANSUBDIRS = man
71 #
72 # MSGSUBDIRS= cmd ucblib ucblib
73 # DOMAINS= \
74 # SUNW_OST_ADMIN \
75 # SUNW_OST_NETRPC \
76 # SUNW_OST_OSCMD \
77 # SUNW_OST_OSLIB \
78 # SUNW_OST_UCBCMD \
79 # SUNW_OST_ZONEINFO
80 #
81 # MSGDDIRS= $(DOMAINS:%=$(MSGROOT)/%)
82 # MSGDIRS= $(MSGROOT) $(MSGDDIRS) $(MSGROOT)/LC_TIME
83 #
84 # all := TARGET= all
85 # install := TARGET= install
86 # install1 := TARGET= install
87 # install2 := TARGET= install
88 # install_h := TARGET= install_h
89 # clean := TARGET= clean
90 # clobber := TARGET= clobber
91 # check := TARGET= check
92 #
93 # .KEEP_STATE:
94 #
95 #
96 # Note: install does not cause a build in pkg. To build packages,
97 # cd pkg and do a 'make install'
98 #
99 #
100 # all: mapfiles closedbins sgs .WAIT $(SUBDIRS) pkg
101 #
102 #
103 # The _msg build is a two-step process. First, the _msg dependency
104 # causes recursive makes in $(MSGSUBDIRS), which stages raw message
105 # files in $(ROOT)/catalog. Second, the action from the install
106 # target rule causes those messages to be post-processed from where
107 # they were staged in $(ROOT)/catalog, and the results placed into the
108 # proto area.
109 #
110 # The stage-licenses target causes the license files needed for
111 # packaging to be pulled from $(SRC) and $(CLOSED) and staged in
112 # $(ROOT)/licenses.
113 #
114 # install: install1 install2 _msg stage-licenses
115 # @cd msg; pwd; $(MAKE) _msg
116 # @rm -rf "$(ROOT)/catalog"
117 #
118 # stage-licenses: install2
119 # @cd pkg; pwd; $(MAKE) stage-licenses
120 #
121 # install1: mapfiles closedbins sgs
```

new/usr/src/Makefile

3

```

123 install2: install1 $(SUBDIRS)

125 _msg: _msgdirs rootdirs FRC
126   @for m in $(MSGSUBDIRS); do \
127     cd $$m; pwd; $(MAKE) _msg; cd ..; \
128   done

130 mapfiles: bldtools
131   @cd common/mapfiles; pwd; $(MAKE) install

133 clean: $(SUBDIRS) head pkg
134 clobber: $(SUBDIRS) head pkg clobber_local
135 clobber_local:
136   @cd tools; pwd; $(MAKE) clobber
137   @cd common/mapfiles; pwd; $(MAKE) clobber
138   @cd msg; pwd; $(MAKE) clobber

140 closedbins: bldtools $(ROOTDIRS) FRC
141   @CLOSED_ROOT="$${ON_CLOSED_BINS}/root_$(MACH)${${RELEASE_BUILD+}-nd}"; \
142   if [ ! -d "$${CLOSED_ROOT}" ]; then \
143     $(ECHO) "Error: ON_CLOSED_BINS must point to closed" \
144     "binaries."; \
145     $(ECHO) "root_$(MACH)${${RELEASE_BUILD+}-nd} is not" \
146     "present in $${ON_CLOSED_BINS}."; \
147     exit 1; \
148   fi; \
149   $(ECHO) "Copying closed binaries from $${CLOSED_ROOT}"; \
150   (cd $${CLOSED_ROOT}; \
151     $(TAR) cfx - $(CODEMGR_WS)/exception_lists/closed-bins .) | \
152     (cd $(ROOT); $(TAR) xBpf -); \
153   ( cd $(ROOT); $(CTFSTRIP) $$(cd $${CLOSED_ROOT}; $(FIND) \
154     ./kernel ./usr/kernel ./platform/*/kernel -type f -a -perm -u+x | \
155     $(EGREP) -vf $(CODEMGR_WS)/exception_lists/closed-bins) )

157 #
158 # Declare what parts can be built in parallel
159 # DUMMY at the end is used in case macro expansion produces an empty string to
160 # prevent everything going in parallel
161 #
162 .PARALLEL: $(PARALLEL_HEADERS) DUMMY
163 .PARALLEL: $(PARALLEL_DIRS) DUMMY

165 $(SUBDIRS) head pkg: FRC
166   @cd $@; pwd; $(MAKE) $(TARGET)

168 # librpcsvc has a dependency on headers installed by
169 # userheaders, hence the .WAIT before libheaders.
170 sgs: rootdirs .WAIT sysheaders userheaders .WAIT \
171   libheaders cmdheaders

173 #
174 # Top-level setup target to setup the development environment that includes
175 # headers, tools and generated mapfiles. For open-only builds (i.e.: source
176 # trees w/o usr/closed), this also depends on the closedbins target (above)
177 # in order to properly seed the proto area. Note, although the tools are
178 # dependent on a number of constant mapfiles, the tools themselves are
179 # required to build the generated mapfiles.
180 #
181 setup: closedbins bldtools sgs mapfiles

183 bldtools:
184   @cd tools; pwd; $(MAKE) install

186 # /var/mail/:saved is a special case because of the colon in the name.
187 #
188 rootdirs: $(ROOTDIRS)

```

new/usr/src/Makefile

4

```

189   $(INS) -d -m 775 $(ROOT)/var/mail/:saved

191 lint: FRC
192   $(MAKE) -f Makefile.lint

194 _msgdirs:   $(MSGDIRS)

196 $(ROOTDIRS) $(MSGDIRS):
197   $(INS.dir)

199 userheaders: FRC
200   @cd head; pwd; $(MAKE) install_h

202 libheaders: bldtools
203   @cd lib; pwd; $(MAKE) install_h

205 sysheaders: FRC
206   @cd uts; pwd; $(MAKE) install_h

208 cmdheaders: FRC
209   @cd cmd/fm; pwd; $(MAKE) install_h
210   @cd cmd/mdb; pwd; $(MAKE) install_h

212 check: $(CHKHDRSUBDIRS) $(CHKMFSTSUBDIRS) $(CHKMANSUBDIRS)
218 check: $(CHKHDRSUBDIRS) $(CHKMFSTSUBDIRS)

214 #
215 # Cross-reference customization: skip all of the subdirectories that
216 # don't contain actual source code.
217 #
218 XRPRUNE = pkg prototypes
219 XRINCDIRS = uts/common head ucbshead

221 cscope.out tags: FRC
222   $(XREF) -f -x $@

224 FRC:

226 #
227 # Targets for reporting compiler versions; nightly uses these.
228 #

230 cc-version:
231   @if $($(MACH)_CC) -_versions >/dev/null 2>/dev/null; then \
232     $(ECHO) 32-bit compiler; \
233     $(ECHO) $($(MACH)_CC); \
234     $($(MACH)_CC) -_versions 2>&1 | \
235     $(EGREP) '^(cw|cc|gcc|primary|shadow)'; \
236   else
237     __COMPILER=`$($(MACH)_CC) -_compiler 2>/dev/null || $(TRUE)`; \
238     if [ -z "$${__COMPILER}" ]; then \
239       $(ECHO) No 32-bit compiler found; \
240       exit 1; \
241     else
242       $(ECHO) 32-bit compiler; \
243       $(ECHO) $($(MACH)_CC); \
244       $(ECHO) $${__COMPILER}; \
245       $($(MACH)_CC) -V 2>&1 | head -1; \
246     fi; \
247   fi

249 cc64-version:
250   @if $($(MACH64)_CC) -_versions >/dev/null 2>/dev/null; then \
251     $(ECHO) 64-bit compiler; \
252     $(ECHO) $($(MACH64)_CC); \
253     $($(MACH64)_CC) -_versions 2>&1 | \

```

```
254         $(EGREP) '^ (cw|cc|gcc|primary|shadow)'; \
255     else \
256         __COMPILER='${$(MACH64)_CC} --compiler 2>/dev/null || $(TRUE)'; \
257     if [ -z "$$__COMPILER" ]; then \
258         $(ECHO) No 64-bit compiler found; \
259         exit 1; \
260     else \
261         $(ECHO) 64-bit compiler; \
262         $(ECHO) ${$(MACH64)_CC}; \
263         $(ECHO) $$__COMPILER; \
264         ${$(MACH64)_CC} -V 2>&1 | head -1; \
265     fi; \
266 fi

268 java-version:
269     @if [ -x "$(JAVAC)" ]; then \
270         $(ECHO) $(JAVAC); \
271         $(JAVA_ROOT)/bin/java -fullversion 2>&1 | head -1; \
272     else \
273         $(ECHO) No Java compiler found; \
274         exit 1; \
275     fi
```

```

*****
10887 Sat Jul 19 19:24:27 2014
new/usr/src/cmd/Makefile
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 # Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
23 # Copyright 2010 Nexenta Systems, Inc. All rights reserved.
24 # Copyright (c) 2012 Joyent, Inc. All rights reserved.
25 # Copyright (c) 2012 by Delphix. All rights reserved.
26 # Copyright (c) 2013 DEY Storage Systems, Inc. All rights reserved.
27 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
28 #
29 include ../Makefile.master
30 #
31 #
32 # Note that the commands 'lp', and 'perl' are first in
33 # Note that the commands 'agents', 'lp', 'perl', and 'man' are first in
34 # the list, violating alphabetical order. This is because they are very
35 # long-running and should be given the most wall-clock time for a
36 # parallel build.
37 #
38 # Commands in the FIRST_SUBDIRS list are built before starting the build
39 # of other commands. Currently this includes only 'isaexec' and
40 # 'platexec'. This is necessary because $(ROOT)/usr/lib/isaexec or
41 # $(ROOT)/usr/lib/platexec must exist when some other commands are built
42 # because their 'make install' creates a hard link to one of them.
43 #
44 # Commands are listed one per line so that TeamWare can auto-merge most
45 # changes.
46 #
47 FIRST_SUBDIRS= \
48 isaexec \
49 platexec
50 #
51 COMMON_SUBDIRS= \
52 allocate \
53 availdevs \
54 lp \
55 perl \

```

```

55 man \
56 Adm \
57 abi \
58 adbgen \
59 acct \
60 acctadm \
61 arch \
62 asa \
63 ast \
64 audio \
65 auths \
66 autopush \
67 avs \
68 awk \
69 awk_xpg4 \
70 backup \
71 banner \
72 bart \
73 basename \
74 bc \
75 bdiff \
76 beadm \
77 bfs \
78 bnu \
79 boot \
80 busstat \
81 cal \
82 calendar \
83 captinfo \
84 cat \
85 cdrw \
86 cfgadm \
87 checkeq \
88 checknr \
89 chgrp \
90 chmod \
91 chown \
92 chroot \
93 clear \
94 clinfo \
95 cmd-crypto \
96 cmd-inet \
97 col \
98 compress \
99 consadm \
100 coreadm \
101 cpio \
102 cpc \
103 cron \
104 crypt \
105 csh \
106 csplit \
107 ctrun \
108 ctstat \
109 ctwatch \
110 datadm \
111 date \
112 dc \
113 dd \
114 deroff \
115 devfsadm \
116 syseventd \
117 devctl \
118 devinfo \
119 devmgmt \
120 devprop \

```

new/usr/src/cmd/Makefile

```

121     dfs.cmds  //
122     diff      //
123     diff3     //
124     diffmk    //
125     dircmp    //
126     dirname   //
127     dis       //
128     diskmgtd  //
129     dispadmin //
130     dladm     //
131     dlstat    //
132     dmesg     //
133     dodatadm  //
134     dtrace    //
135     du        //
136     dumpadm   //
137     dumpcs    //
138     echo      //
139     ed        //
140     eeprom    //
141     egrep     //
142     eject     //
143     emul64ioctl //
144     enhance   //
145     env       //
146     eqn       //
147     expand     //
148     expr      //
149     exstr     //
150     factor    //
151     false     //
152     fcinfo    //
153     fcoesvc   //
154     fdetach   //
155     fdformat  //
156     fdisk     //
157     filesync  //
158     fgrep     //
159     file      //
160     find      //
161     flowadm   //
162     flowstat  //
163     fm        //
164     fmt       //
165     fmthard   //
166     fmtmsg    //
167     fold      //
168     format    //
169     fs.d      //
170     fstyp     //
171     fuser     //
172     fwflash   //
173     gcore     //
174     gencat    //
175     geniconvtbl //
176     genmsg    //
177     getconf   //
178     getdevpolicy //
179     getent    //
180     getfacl   //
181     getmajor  //
182     getopt    //
183     gettext   //
184     gettxt    //
185     grep      //
186     grep_xpg4 //

```

3

new/usr/src/cmd/Makefile

```

187     groups   //
188     grpck    //
189     gss      //
190     hal      //
191     halt     //
192     head     //
193     hostid   //
194     hostname //
195     hotplug  //
196     hotplugd //
197     hwdata   //
198     ibd_upgrade //
199     id       //
200     idmap    //
201     infocmp  //
202     init     //
203     initpkg  //
204     install.d //
205     intrd    //
206     intrstat //
207     ipcrm    //
208     ipcs     //
209     ipdadm   //
210     ipf      //
211     isainfo  //
212     isalist  //
213     itutools //
214     iscsiadm //
215     iscsid   //
216     iscsitsvc //
217     isns     //
218     itadm    //
219     java     //
220     kbd      //
221     keyserv  //
222     killall  //
223     krb5     //
224     ksh      //
225     kvmstat  //
226     last     //
227     lastcomm //
228     latencytop //
229     ldap     //
230     ldapcachemgr //
231     lgrpinfo //
232     line     //
233     link     //
234     dlmgtd   //
235     listen   //
236     loadkeys //
237     locale   //
238     localedef //
239     lockstat //
240     locator  //
241     lofiadm  //
242     logadm   //
243     logger   //
244     login    //
245     logins   //
246     look     //
247     ls       //
248     luxadm   //
249     lvm      //
250     mach     //
251     machid   //
252     mail     //

```

4

```

253 mailx //
254 makekey //
255 man //
256 mandoc //
257 mdb //
258 mesg //
259 mkdir //
260 mkfifo //
261 mkfile //
262 mkmsgs //
263 mknod //
264 mkpwdict //
265 mktemp //
266 modload //
267 more //
268 mpathadm //
269 msgfmt //
270 msgid //
271 mt //
272 mv //
273 mvdir //
274 ndmpadm //
275 ndmpd //
276 ndmpstat //
277 netadm //
278 netfiles //
279 newform //
280 newgrp //
281 news //
282 newtask //
283 nice //
284 nl //
285 nlsadmin //
286 nohup //
287 nsadmin //
288 nscd //
289 oamuser //
290 oawk //
291 od //
292 pack //
293 pagesize //
294 passmgmt //
295 passwd //
296 pathchk //
297 pbind //
298 pcidr //
299 pcitool //
300 pexec //
301 pexecd //
302 pginfo //
303 pgstat //
304 pgrep //
305 picl //
306 plimit //
307 policykit //
308 pools //
309 power //
310 powertop //
311 ppgsz //
312 pg //
313 plockstat //
314 pr //
315 prctl //
316 print //
317 printf //
318 priocntl //

```

```

319 profiles //
320 projadd //
321 projects //
322 prstat //
323 prtconf //
324 prtdiag //
325 prvtoc //
326 ps //
327 psradm //
328 psrinfo //
329 psrset //
330 ptools //
331 pwck //
332 pwconv //
333 pwd //
334 pyzfs //
335 raidctl //
336 ramdiskadm //
337 rcap //
338 rcm_daemon //
339 rctladm //
340 refer //
341 regcmp //
342 renice //
343 rexd //
344 rm //
345 rmdir //
346 rmformat //
347 rmmount //
348 rmt //
349 rmvolmgr //
350 roles //
351 rpcbind //
352 rpcgen //
353 rpcinfo //
354 rpcsvc //
355 runat //
356 sa //
357 saf //
358 sasinfo //
359 savecore //
360 sbdadm //
361 script //
362 scsi //
363 sdiff //
364 sdpadm //
365 sed //
366 sendmail //
367 setfacl //
368 setmnt //
369 setpgrp //
370 setuname //
371 sgs //
372 sh //
373 shcomp //
374 smbios //
375 smbstrv //
376 smserverd //
377 soelim //
378 sort //
379 spell //
380 split //
381 sqlite //
382 srchtxt //
383 srptadm //
384 srptsvc //

```


new/usr/src/cmd/Makefile

```

385      ssh
386      stat
387      stmfadm
388      stmfproxy
389      stmfsvc
390      stmsboot
391      streams
392      strings
393      su
394      sulogin
395      sunpc
396      svc
397      svr4pkg
398      swap
399      sync
400      sysdef
401      syseventadm
402      syslogd
403      tabs
404      tail
405      tar
406      tbl
407      tcopy
408      tcpd
409      terminfo
410      th_tools
411      tic
412      time
413      tip
414      tnf
415      touch
416      tput
417      tr
418      trapstat
419      troff
420      true
421      truss
422      tsol
423      tty
424      ttymon
425      tzreload
426      uadmin
427      ul
428      uname
429      units
430      unlink
431      unpack
432      userattr
433      users
434      utmp_update
435      utmpd
436      valtools
437      vgrind
438      vi
439      volcheck
440      volrmmount
441      vrrpadm
442      vscan
443      vt
444      w
445      wall
446      which
447      who
448      whodo
449      wracct
450      write

```

7

new/usr/src/cmd/Makefile

```

451      wusbadm
452      xargs
453      xstr
454      yes
455      ypcmd
456      yppasswd
457      zdb
458      zdump
459      zfs
460      zhack
461      zic
462      zinject
463      zlogin
464      zoneadm
465      zoneadmd
466      zonecfg
467      zonename
468      zpool
469      zlook
470      zonestat
471      zstreamdump
472      ztest

474 i386_SUBDIRS=
475      acpihp
476      addbadsec
477      biosdev
478      diskscan
479      lms
480      ntfsprogs
481      parted
482      rtc
483      ucodeadm
484      xvm

486 sparc_SUBDIRS=
487      cvcd
488      dcs
489      device_remap
490      drd
491      fruadm
492      ldmad
493      oplhp
494      prtdscp
495      prtfru
496      scadm
497      sckmd
498      sf880drd
499      virtinfo
500      vntsd

502 #
503 # Commands that are messaged. Note that 'lp' comes first
504 # (see previous comment about 'lp'.)
505 # Commands that are messaged. Note that 'lp' and 'man' come first
506 # (see previous comment about 'lp' and 'man').
507 #
508 MSGSUBDIRS=
509      lp
510      man
511      abi
512      acctadm
513      allocate
514      asa
515      audio
516      audit

```

8

new/usr/src/cmd/Makefile

```

514     auditconfig //
515     auditd //
516     auditrecord //
517     auditset //
518     auths //
519     autopush //
520     avs //
521     awk //
522     awk_xpg4 //
523     backup //
524     banner //
525     bart //
526     basename //
527     beadm //
528     bnu //
529     busstat //
530     cal //
531     cat //
532     cdrw //
533     cfgadm //
534     checkeq //
535     checknr //
536     chgrp //
537     chmod //
538     chown //
539     cmd-crypto //
540     cmd-inet //
541     col //
542     compress //
543     consadm //
544     coreadm //
545     cpio //
546     cpc //
547     cron //
548     csh //
549     csplit //
550     ctrun //
551     ctstat //
552     ctwatch //
553     datadm //
554     date //
555     dc //
556     dcs //
557     dd //
558     deroff //
559     devfsadm //
560     dfs.cmds //
561     diff //
562     diffmk //
563     dladm //
564     dlstat //
565     du //
566     dumpcs //
567     ed //
568     eject //
569     env //
570     eqn //
571     expand //
572     expr //
573     fcinfo //
574     fgrep //
575     file //
576     filesync //
577     find //
578     flowadm //
579     flowstat //

```

9

new/usr/src/cmd/Makefile

```

580     fm //
581     fold //
582     fs.d //
583     fwflash //
584     geniconvtbl //
585     genmsg //
586     getconf //
587     getent //
588     gettext //
589     gettxt //
590     grep //
591     grep_xpg4 //
592     grpck //
593     gss //
594     halt //
595     head //
596     hostname //
597     hotplug //
598     id //
599     idmap //
600     isaexec //
601     iscsiadm //
602     iscsid //
603     isns //
604     itadm //
605     kbd //
606     krb5 //
607     ksh //
608     last //
609     ldap //
610     ldapcachemgr //
611     lgrpinfo //
612     locale //
613     lofiadm //
614     logadm //
615     logger //
616     logins //
617     ls //
618     luxadm //
619     lvm //
620     mailx //
621     man //
622     mesg //
623     mkdir //
624     mkpwdict //
625     mktemp //
626     more //
627     mpathadm //
628     msgfmt //
629     mv //
630     ndmpadm //
631     ndmpstat //
632     newgrp //
633     newtask //
634     nice //
635     nohup //
636     oawk //
637     pack //
638     passwd //
639     passmgmt //
640     pathchk //
641     pexec //
642     pg //
643     pgrep //
644     picl //
645     pools //

```

10

```

646 power          \
647 pr             \
648 praudit        \
649 print          \
650 profiles       \
651 projadd        \
652 projects       \
653 prstat         \
654 prtdiag        \
655 ps             \
656 psrinfo        \
657 ptools         \
658 pwconv         \
659 pwd            \
660 pyzfs          \
661 raidctl        \
662 ramdiskadm     \
663 rcap           \
664 rcm_daemon     \
665 refer          \
666 regcmp         \
667 renice         \
668 roles          \
669 rm             \
670 rmdir          \
671 rmformat       \
672 rmmount        \
673 rmvolmgr       \
674 sasinfo        \
675 sbdadm         \
676 scadm          \
677 script         \
678 scsi           \
679 sdiff          \
680 sdpadm         \
681 sgs            \
682 sh             \
683 shcomp         \
684 smbstrv        \
685 sort           \
686 split          \
687 srptadm        \
688 ssh            \
689 stat           \
690 stmfadm        \
691 stmsboot       \
692 strings        \
693 su             \
694 svc            \
695 svr4pkg        \
696 swap           \
697 syseventadm    \
698 syseventd      \
699 tabs           \
700 tar            \
701 tbl            \
702 time           \
703 tnf            \
704 touch          \
705 tput           \
706 troff          \
707 tsol           \
708 tty            \
709 ttymon         \
710 tzreload       \
711 ul             \

```

```

712 uname          \
713 units          \
714 unlink         \
715 unpack         \
716 userattr       \
717 valtools       \
718 vgrind         \
719 vi             \
720 volcheck       \
721 volrmmount     \
722 vrrpdm         \
723 vsca           \
724 w              \
725 who            \
726 whodo          \
727 wracct         \
728 write          \
729 wusbadm        \
730 xargs          \
731 yppasswd       \
732 zdump          \
733 zfs            \
734 zic            \
735 zlogin         \
736 zoneadm        \
737 zoneadm        \
738 zonecfg        \
739 zonename       \
740 zpool          \
741 zonestat       \
743 sparc_MSGSUBDIRS= \
744 fruadm         \
745 prttdscp       \
746 prtfru         \
747 virtinfo       \
748 vntsd          \
750 i386_MSGSUBDIRS= \
751 ucodeadm       \
753 #              \
754 # commands that use dcgettext for localized time, LC_TIME
755 #              \
756 DCSUBDIRS=     \
757 cal            \
758 cfgadm         \
759 diff           \
760 ls             \
761 pr             \
762 ps             \
763 tar            \
764 w              \
765 who            \
766 whodo          \
767 write          \
769 #              \
770 # commands that belong only to audit.
771 #              \
772 AUDITSUBDIRS=  \
773 amt            \
774 audit          \
775 audit_warn     \
776 auditconfig    \
777 auditd         \

```

new/usr/src/cmd/Makefile

13

```
778      auditrecord \
779      auditreduce \
780      auditset \
781      auditstat \
782      praudit

784 #
785 # commands not owned by the systems group
786 #
787 BWOSDIRS=

790 all :=          TARGET = all
791 install :=      TARGET = install
792 clean :=       TARGET = clean
793 clobber :=     TARGET = clobber
794 lint :=        TARGET = lint
795 _msg :=        TARGET = _msg
796 _dc :=         TARGET = _dc

798 .KEEP_STATE:

800 SUBDIRS = $(COMMON_SUBDIRS) $($ (MACH)_SUBDIRS)

802 .PARALLEL:     $(BWOSDIRS) $(SUBDIRS) $(MSGSUBDIRS) $(AUDITSUBDIRS)

804 all install clean clobber lint: $(FIRST_SUBDIRS) .WAIT $(SUBDIRS) \
805      $(AUDITSUBDIRS)

807 #
808 # Manifests cannot be checked in parallel, because we are using
809 # the global repository that is in $(SRC)/cmd/svc/seed/global.db.
810 # For this reason, to avoid .PARALLEL and .NO_PARALLEL conflicts,
811 # we spawn off a sub-make to perform the non-parallel 'make check'
812 #
813 check:
814      $(MAKE) -f Makefile.check check

816 #
817 # The .WAIT directive works around an apparent bug in parallel make.
818 # Evidently make was getting the target _msg vs. _dc confused under
819 # some level of parallelization, causing some of the _dc objects
820 # not to be built.
821 #
822 _msg: $(MSGSUBDIRS) $($ (MACH)_MSGSUBDIRS) .WAIT _dc

824 _dc: $(DCSUBDIRS)

826 #
827 # Dependencies
828 #
829 fs.d: fstyp
830 ksh:   shcomp isaexec
831 mdb:  terminfo
832 print: lp

834 $(FIRST_SUBDIRS) $(BWOSDIRS) $(SUBDIRS) $(AUDITSUBDIRS): FRC
835     @if [ -f $@/Makefile ]; then \
836         cd $@; pwd; $(MAKE) $(TARGET); \
837     else \
838         true; \
839     fi

841 FRC:
```

```
*****
```

```
1047 Sat Jul 19 19:24:27 2014
```

```
new/usr/src/cmd/man/Makefile
```

```
5025 import and use mandoc
```

```
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
```

```
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
```

```
Reviewed by: Robert Mustacchi <rm@joyent.com>
```

```
Reviewed by: Albert Lee <trisk@nexenta.com>
```

```
Approved by: TBD
```

```
*****
```

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # CDDL HEADER START
8 #
9 # A full copy of the text of the CDDL should have accompanied this
10 # source. A copy of the CDDL is also available via the Internet at
11 # http://www.illumos.org/license/CDDL.
12 # The contents of this file are subject to the terms of the
13 # Common Development and Distribution License, Version 1.0 only
14 # (the "License"). You may not use this file except in compliance
15 # with the License.
16 #
17 #
18 #
19 #
20 #
21 #
22 #
23 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
24 # or http://www.opensolaris.org/os/licensing.
25 # See the License for the specific language governing permissions
26 # and limitations under the License.
27 #
28 #
29 # Copyright 2012 Nexenta Systems, Inc. All rights reserved.
30 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
31 # When distributing Covered Code, include this CDDL HEADER in each
32 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
33 # If applicable, add the following below this CDDL HEADER, with the
34 # fields enclosed by brackets "[]" replaced with your own identifying
35 # information: Portions Copyright [yyyy] [name of copyright owner]
36 #
37 #
38 #
39 # CDDL HEADER END
40 #
41 #
42 #
43 #ident "%Z%M% %I% %E% SMI"
44 #
45 # Copyright (c) 1990 by Sun Microsystems, Inc.
46 #
47 #
48 # cmd/man/Makefile
```

```
17 PROG= man
18 LINKS= apropos whatis catman
19 LIBLINKS = makewhatis
20 OBJS= makewhatis.o man.o stringlist.o
21 SRCS= $(OBJS:%.o=%.c)
22 include ../Makefile.cmd
23 SUBDIRS = src
```

```
23 include $(SRC)/cmd/Makefile.cmd
24 all := TARGET= all
25 install := TARGET= install
26 clean := TARGET= clean
27 clobber := TARGET= clobber
28 lint := TARGET= lint
29 _msg := TARGET= catalog
```

```
25 CFLAGS += $(CCVERBOSE)
26 #for message catalog files
```

```
40 POFILE = man.po
```

```
41 POFILES = src/src.po
```

```
27 ROOTLINKS= $(LINKS:%=$(ROOTBIN)/%) $(LIBLINKS:%=$(ROOTLIB)/%)
```

```
43 .KEEP_STATE:
```

```
29 .KEEP_STATE :
```

```
45 all install clean lint: $(SUBDIRS)
```

```
31 all: $(PROG)
```

```
47 clobber: $(SUBDIRS) local_clobber
```

```
33 clean:
```

```
34 $(RM) $(OBJS)
```

```
49 local_clobber:
```

```
50 $(RM) $(CLOBBERFILES)
```

```
36 install: all $(ROOTPROG) $(ROOTLINKS)
```

```
52 _msg: $(SUBDIRS)
```

```
53 $(RM) $(POFILE)
```

```
54 cat $(POFILES) > $(POFILE)
```

```
55 $(RM) $(MSGDOMAIN)/$(POFILE)
```

```
56 cp $(POFILE) $(MSGDOMAIN)
```

```
38 lint: lint_SRCS
```

```
58 $(SUBDIRS): FRC
```

```
59 @cd $@; pwd; $(MAKE) $(TARGET)
```

```
40 $(PROG): $(OBJS)
```

```
41 $(LINK.c) $(OBJS) -o $@ $(LDLIBS)
```

```
42 $(POST_PROCESS)
```

```
44 $(ROOTLINKS): $(ROOTPROG)
```

```
45 $(RM) $@; $(LN) $(ROOTPROG) $@
```

```
47 include $(SRC)/cmd/Makefile.targ
```

```
61 FRC:
```

 4680 Sat Jul 19 19:24:27 2014
 new/usr/src/cmd/man/THIRDPARTYLICENSE
 5025 import and use mandoc
 Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
 Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
 Reviewed by: Robert Mustacchi <rm@joyent.com>
 Reviewed by: Albert Lee <trisk@nexenta.com>
 Approved by: TBD

```

1 man.c:
3 Copyright (c) 1980 Regents of the University of California.
4 All rights reserved.

6 Redistribution and use in source and binary forms, with or without
7 modification, are permitted provided that the following conditions are
8 met:
10 1. Redistributions of source code must retain the above copyright
11 notice, this list of conditions and the following disclaimer.
12 2. Redistributions in binary form must reproduce the above
13 copyright notice, this list of conditions and the following
14 disclaimer in the documentation and/or other materials provided
15 with the distribution.
16 3. All advertising materials mentioning features or use of this
17 software must display the following acknowledgement:
18     This product includes software developed by the University
19     of California, Berkeley and its contributors.
20 4. Neither the name of the University nor the names of its
21 contributors may be used to endorse or promote products derived
22 from this software without specific prior written permission.

24 THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND
25 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
26 IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
27 PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
28 CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
29 EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
30 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
31 PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
32 LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
33 NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
34 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

37 makewhatis.c:
39 Copyright (c) 2002 John Rochester
40 All rights reserved.

42 Redistribution and use in source and binary forms, with or without
43 modification, are permitted provided that the following conditions
44 are met:
45 1. Redistributions of source code must retain the above copyright
46 notice, this list of conditions and the following disclaimer,
47 in this position and unchanged.
48 2. Redistributions in binary form must reproduce the above copyright
49 notice, this list of conditions and the following disclaimer in the
50 documentation and/or other materials provided with the distribution.
51 3. The name of the author may not be used to endorse or promote products
52 derived from this software without specific prior written permission

54 THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR
55 IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
56 OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.

```

```

57 IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
58 INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
59 NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
60 DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
61 THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
62 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
63 THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

66 stringlist.c, stringlist.h:

68 Copyright (c) 1994 Christos Zoulas
69 All rights reserved.

71 Redistribution and use in source and binary forms, with or without
72 modification, are permitted provided that the following conditions
73 are met:
74 1. Redistributions of source code must retain the above copyright
75 notice, this list of conditions and the following disclaimer.
76 2. Redistributions in binary form must reproduce the above copyright
77 notice, this list of conditions and the following disclaimer in the
78 documentation and/or other materials provided with the distribution.
79 4. The name of the author may not be used to endorse or promote products
80 derived from this software without specific prior written permission.

82 THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS
83 OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
84 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
85 ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
86 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
87 DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
88 OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
89 HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
90 LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
91 OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
92 SUCH DAMAGE.

```

new/usr/src/cmd/man/makewhatis.c

1

18255 Sat Jul 19 19:24:27 2014

new/usr/src/cmd/man/makewhatis.c

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /*
2  * Copyright (c) 2002 John Rochester
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions
7  * are met:
8  * 1. Redistributions of source code must retain the above copyright
9  * notice, this list of conditions and the following disclaimer,
10 * in this position and unchanged.
11 * 2. Redistributions in binary form must reproduce the above copyright
12 * notice, this list of conditions and the following disclaimer in the
13 * documentation and/or other materials provided with the distribution.
14 * 3. The name of the author may not be used to endorse or promote products
15 * derived from this software without specific prior written permission
16 *
17 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
18 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
19 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
20 * IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
21 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
22 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
23 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
24 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
25 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
26 * THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
27 */
28
29 /*
30 * Copyright 2012 Nexenta Systems, Inc. All rights reserved.
31 * Copyright 2014 Garrett D'Amore <garrett@damore.org>
32 */
33
34 #include <sys/types.h>
35 #include <sys/stat.h>
36 #include <sys/param.h>
37
38 #include <ctype.h>
39 #include <dirent.h>
40 #include <err.h>
41 #include <signal.h>
42 #include <stddef.h>
43 #include <stdio.h>
44 #include <stdlib.h>
45 #include <string.h>
46 #include <unistd.h>
47
48 #include "man.h"
49 #include "stringlist.h"
50
51 /* Information collected about each man page in a section */
52 struct page_info {
53     char    *filename;
54     char    *name;
55     char    *suffix;
```

new/usr/src/cmd/man/makewhatis.c

2

```
57     ino_t    inode;
58 };
59
60 /* An expanding string */
61 struct sbuf {
62     char    *content; /* the start of the buffer */
63     char    *end; /* just past the end of the content */
64     char    *last; /* the last allocated character */
65 };
66
67 /* Remove the last amount characters from the sbuf */
68 #define sbuf_retract(sbuf, amount) ((sbuf)->end -= (amount))
69 /* Return the length of the sbuf content */
70 #define sbuf_length(sbuf) ((sbuf)->end - (sbuf)->content)
71
72 typedef char *edited_copy(char *from, char *to, int length);
73
74 /*
75  * While the whatis line is being formed, it is stored in whatis_proto.
76  * When finished, it is reformatted into whatis_final and then appended
77  * to whatis_lines.
78  */
79 static struct sbuf    *whatis_proto;
80 static struct sbuf    *whatis_final;
81 static stringlist    *whatis_lines; /* collected output lines */
82
83 static char tempfile[MAXPATHLEN]; /* path of temporary file, if any */
84
85 #define MDOC_COMMANDS "ArDvErEvFlLiNmPa"
86
87
88 /* Free a struct page_info and its content */
89 static void
90 free_page_info(struct page_info *info)
91 {
92     free(info->filename);
93     free(info->name);
94     free(info->suffix);
95     free(info);
96 }
97
98
99 /*
100 * Allocate and fill in a new struct page_info given the
101 * name of the man section directory and the dirent of the file.
102 * If the file is not a man page, return NULL.
103 */
104 static struct page_info *
105 new_page_info(char *dir, struct dirent *dirent)
106 {
107     struct page_info *info;
108     int    basename_length;
109     char    *suffix;
110     struct stat    st;
111
112     if ((info = malloc(sizeof (struct page_info))) == NULL)
113         err(1, "malloc");
114     basename_length = strlen(dirent->d_name);
115     suffix = &dirent->d_name[basename_length];
116     if (asprintf(&info->filename, "%s/%s", dir, dirent->d_name) == -1)
117         err(1, "asprintf");
118     for (;;) {
119         if (--suffix == dirent->d_name || !isalnum(*suffix)) {
120             if (*suffix == '.')
121                 break;
122             free(info->filename);
```

```

123         free(info);
124         return (NULL);
125     }
126 }
127 *suffix++ = '\0';
128 info->name = strdup(dirent->d_name);
129 info->suffix = strdup(suffix);
130 if (stat(info->filename, &st) < 0) {
131     warn("%s", info->filename);
132     free_page_info(info);
133     return (NULL);
134 }
135 if (!S_ISREG(st.st_mode)) {
136     free_page_info(info);
137     return (NULL);
138 }
139 info->inode = st.st_ino;
140 return (info);
141 }
142
143 /*
144  * Reset sbuf length to 0.
145  */
146 static void
147 sbuf_clear(struct sbuf *sbuf)
148 {
149     sbuf->end = sbuf->content;
150 }
151
152
153 /*
154  * Allocate a new sbuf.
155  */
156 static struct sbuf *
157 new_sbuf(void)
158 {
159     struct sbuf *sbuf;
160
161     if ((sbuf = malloc(sizeof (struct sbuf))) == NULL)
162         err(1, "malloc");
163     if ((sbuf->content = (char *)malloc(LINE_ALLOC)) == NULL)
164         err(1, "malloc");
165     sbuf->last = sbuf->content + LINE_ALLOC - 1;
166     sbuf_clear(sbuf);
167
168     return (sbuf);
169 }
170
171 /*
172  * Ensure that there is enough room in the sbuf
173  * for nchars more characters.
174  */
175 static void
176 sbuf_need(struct sbuf *sbuf, int nchars)
177 {
178     char *new_content;
179     size_t size, cntsize;
180     size_t grow = 128;
181
182     while (grow < nchars) {
183         grow += 128; /* we grow in chunks of 128 bytes */
184     }
185
186     /* Grow if the buffer isn't big enough */
187     if (sbuf->end + nchars > sbuf->last) {
188         size = sbuf->last + 1 - sbuf->content;

```

```

189         size += grow;
190         cntsize = sbuf->end - sbuf->content;
191
192         if ((new_content = realloc(sbuf->content, size)) == NULL) {
193             perror("realloc");
194             if (tempfile[0] != '\0')
195                 (void) unlink(tempfile);
196             exit(1);
197         }
198         sbuf->content = new_content;
199         sbuf->end = new_content + cntsize;
200         sbuf->last = new_content + size - 1;
201     }
202 }
203
204 /*
205  * Append a string of a given length to the sbuf.
206  */
207 static void
208 sbuf_append(struct sbuf *sbuf, const char *text, int length)
209 {
210     if (length > 0) {
211         sbuf_need(sbuf, length);
212         (void) memcpy(sbuf->end, text, length);
213         sbuf->end += length;
214     }
215 }
216
217 /*
218  * Append a null-terminated string to the sbuf.
219  */
220 static void
221 sbuf_append_str(struct sbuf *sbuf, char *text)
222 {
223     sbuf_append(sbuf, text, strlen(text));
224 }
225
226 /*
227  * Append an edited null-terminated string to the sbuf.
228  */
229 static void
230 sbuf_append_edited(struct sbuf *sbuf, char *text, edited_copy copy)
231 {
232     int length;
233
234     if ((length = strlen(text)) > 0) {
235         sbuf_need(sbuf, length);
236         sbuf->end = copy(text, sbuf->end, length);
237     }
238 }
239
240 /*
241  * Strip any of a set of chars from the end of the sbuf.
242  */
243 static void
244 sbuf_strip(struct sbuf *sbuf, const char *set)
245 {
246     while (sbuf->end > sbuf->content && strchr(set, sbuf->end[-1]) != NULL)
247         sbuf->end--;
248 }
249
250 /*
251  * Return the null-terminated string built by the sbuf.
252  */

```



```

255 static char *
256 sbuf_content(struct sbuf *sbuf)
257 {
258
259     *sbuf->end = '\0';
260     return (sbuf->content);
261 }
262
263 /*
264 * Return true if no man page exists in the directory with
265 * any of the names in the stringlist.
266 */
267 static int
268 no_page_exists(char *dir, stringlist *names, char *suffix)
269 {
270     char    path[MAXPATHLEN];
271     char    *suffixes[] = { "", ".gz", ".bz2", NULL };
272     size_t  i;
273     int     j;
274
275     for (i = 0; i < names->sl_cur; i++) {
276         for (j = 0; suffixes[j] != NULL; j++) {
277             (void) snprintf(path, MAXPATHLEN, "%s/%s.%s%s",
278                 dir, names->sl_str[i], suffix, suffixes[j]);
279             if (access(path, F_OK) == 0) {
280                 return (0);
281             }
282         }
283     }
284     return (1);
285 }
286
287 /* ARGSUSED sig */
288 static void
289 trap_signal(int sig)
290 {
291
292     if (tempfile[0] != '\0')
293         (void) unlink(tempfile);
294
295     exit(1);
296 }
297
298 /*
299 * Attempt to open an output file.
300 * Return NULL if unsuccessful.
301 */
302 static FILE *
303 open_output(char *name)
304 {
305     FILE    *output;
306
307     whatis_lines = sl_init();
308     (void) snprintf(tempfile, MAXPATHLEN, "%s.tmp", name);
309     name = tempfile;
310     if ((output = fopen(name, "w")) == NULL) {
311         warn("%s", name);
312         return (NULL);
313     }
314     return (output);
315 }
316
317 static int
318 linesort(const void *a, const void *b)
319 {

```

```

321     return (strcmp((*(const char * const *)a), (*(const char * const *)b)));
322 }
323
324 /*
325 * Write the unique sorted lines to the output file.
326 */
327 static void
328 finish_output(FILE *output, char *name)
329 {
330     size_t  i;
331     char    *prev = NULL;
332
333     qsort(whatis_lines->sl_str, whatis_lines->sl_cur, sizeof (char *),
334         linesort);
335     for (i = 0; i < whatis_lines->sl_cur; i++) {
336         char *line = whatis_lines->sl_str[i];
337         if (i > 0 && strcmp(line, prev) == 0)
338             continue;
339         prev = line;
340         (void) fputs(line, output);
341         (void) putc('\n', output);
342     }
343     (void) fclose(output);
344     sl_free(whatis_lines, 1);
345     (void) rename(tempfile, name);
346     (void) unlink(tempfile);
347 }
348
349 static FILE *
350 open_whatis(char *mandir)
351 {
352     char    filename[MAXPATHLEN];
353
354     (void) snprintf(filename, MAXPATHLEN, "%s/%s", mandir, WHATIS);
355     return (open_output(filename));
356 }
357
358 static void
359 finish_whatis(FILE *output, char *mandir)
360 {
361     char    filename[MAXPATHLEN];
362
363     (void) snprintf(filename, MAXPATHLEN, "%s/%s", mandir, WHATIS);
364     finish_output(output, filename);
365 }
366
367 /*
368 * Remove trailing spaces from a string, returning a pointer to just
369 * beyond the new last character.
370 */
371 static char *
372 trim_rhs(char *str)
373 {
374     char    *rhs;
375
376     rhs = &str[strlen(str)];
377     while (--rhs > str && isspace(*rhs))
378         ;
379     *++rhs = '\0';
380     return (rhs);
381 }
382
383 /*
384 * Return a pointer to the next non-space character in the string.
385 */
386 static char *

```

```

387 skip_spaces(char *s)
388 {
390     while (*s != '\0' && isspace(*s))
391         s++;
393     return (s);
394 }
396 /*
397 * Return whether the line is of one of the forms:
398 * .Sh NAME
399 * .Sh "NAME"
400 * etc.
401 * assuming that section_start is ".Sh".
402 */
403 static int
404 name_section_line(char *line, const char *section_start)
405 {
406     char        *rhs;
408     if (strncmp(line, section_start, 3) != 0)
409         return (0);
410     line = skip_spaces(line + 3);
411     rhs = trim_rhs(line);
412     if (*line == "'") {
413         line++;
414         if (*--rhs == "'")
415             *rhs = '\0';
416     }
417     if (strcmp(line, "NAME") == 0)
418         return (1);
420     return (0);
421 }
423 /*
424 * Copy characters while removing the most common nroff/troff markup:
425 * \em, \mi, \s[+-N], \&
426 * \fF, \f(fo, \f[font]
427 * \*s, \*(st, \*[stringvar]
428 */
429 static char *
430 de_nroff_copy(char *from, char *to, int fromlen)
431 {
432     char        *from_end = &from[fromlen];
434     while (from < from_end) {
435         switch (*from) {
436             case '\\':
437                 switch (*++from) {
438                     case ' ':
439                         if (strncmp(&from[1], "em", 2) == 0 ||
440                             strncmp(&from[1], "mi", 2) == 0) {
441                             from += 3;
442                             continue;
443                         }
444                         break;
445                     case 's':
446                         if (*++from == '-')
447                             from++;
448                         while (isdigit(*from))
449                             from++;
450                         continue;
451                     case 'f':
452                     case '*':

```

```

453         if (*++from == '(') {
454             from += 3;
455         } else if (*from == '[') {
456             while (*++from != ']') &&
457                 from < from_end)
458                 ;
459             from++;
460         } else {
461             from++;
462         }
463         continue;
464     case '&':
465         from++;
466         continue;
467     }
468     break;
469 }
470 *to++ = *from++;
471 }
472 return (to);
473 }
475 /*
476 * Append a string with the nroff formatting removed.
477 */
478 static void
479 add_nroff(char *text)
480 {
482     sbuf_append_edited(whatis_proto, text, de_nroff_copy);
483 }
485 /*
486 * Appends "name(suffix), " to whatis_final
487 */
488 static void
489 add_whatis_name(char *name, char *suffix)
490 {
492     if (*name != '\0') {
493         sbuf_append_str(whatis_final, name);
494         sbuf_append(whatis_final, "(", 1);
495         sbuf_append_str(whatis_final, suffix);
496         sbuf_append(whatis_final, ")", 3);
497     }
498 }
500 /*
501 * Processes an old-style man(7) line. This ignores commands with only
502 * a single number argument.
503 */
504 static void
505 process_man_line(char *line)
506 {
507     char        *p;
509     if (*line == '.') {
510         while (isalpha(++line))
511             ;
512         p = line = skip_spaces(line);
513         while (*p != '\0') {
514             if (!isdigit(*p))
515                 break;
516             p++;
517         }
518         if (*p == '\0')

```

```

519         return;
520     } else
521         line = skip_spaces(line);
522     if (*line != '\0') {
523         add_nroff(line);
524         sbuf_append(whatis_proto, " ", 1);
525     }
526 }

528 /*
529  * Processes a new-style mdoc(7) line.
530 */
531 static void
532 process_mdoc_line(char *line)
533 {
534     int     xref;
535     int     arg = 0;
536     char    *line_end = &line[strlen(line)];
537     int     orig_length = sbuf_length(whatis_proto);
538     char    *next;

540     if (*line == '\0')
541         return;
542     if (line[0] != '.' || !isupper(line[1]) || !islower(line[2])) {
543         add_nroff(skip_spaces(line));
544         sbuf_append(whatis_proto, " ", 1);
545         return;
546     }
547     xref = strncmp(line, ".Xr", 3) == 0;
548     line += 3;
549     while ((line = skip_spaces(line)) < line_end) {
550         if (*line == '"') {
551             next = ++line;
552             for (;;) {
553                 next = strchr(next, '"');
554                 if (next == NULL)
555                     break;
556                 (void) memmove(next, next + 1, strlen(next));
557                 line_end--;
558                 if (*next != '"')
559                     break;
560                 next++;
561             }
562         } else {
563             next = strpbrk(line, "\t");
564         }
565         if (next != NULL)
566             *next++ = '\0';
567         else
568             next = line_end;
569         if (isupper(*line) && islower(line[1]) && line[2] == '\0') {
570             if (strcmp(line, "Ns") == 0) {
571                 arg = 0;
572                 line = next;
573                 continue;
574             }
575             if (strstr(line, MDOC_COMMANDS) != NULL) {
576                 line = next;
577                 continue;
578             }
579         }
580         if (arg > 0 && strchr(",.:;!)", *line) == 0) {
581             if (xref) {
582                 sbuf_append(whatis_proto, "(", 1);
583                 add_nroff(line);
584                 sbuf_append(whatis_proto, ")", 1);

```

```

585         xref = 0;
586         } else {
587             sbuf_append(whatis_proto, " ", 1);
588         }
589     }
590     add_nroff(line);
591     arg++;
592     line = next;
593 }
594     if (sbuf_length(whatis_proto) > orig_length)
595         sbuf_append(whatis_proto, " ", 1);
596 }

598 /*
599  * Collect a list of comma-separated names from the text.
600 */
601 static void
602 collect_names(stringlist *names, char *text)
603 {
604     char    *arg;

606     for (;;) {
607         arg = text;
608         text = strchr(text, ',');
609         if (text != NULL)
610             *text++ = '\0';
611         (void) sl_add(names, arg);
612         if (text == NULL)
613             return;
614         if (*text == ' ')
615             text++;
616     }
617 }

619 enum { STATE_UNKNOWN, STATE_MANSTYLE, STATE_MDOCNAME, STATE_MDOCDESC };

621 /*
622  * Process a man page source into a single whatis line and add it
623  * to whatis_lines.
624 */
625 static void
626 process_page(struct page_info *page, char *section_dir)
627 {
628     FILE    *fp;
629     stringlist *names;
630     char    *descr;
631     int     state = STATE_UNKNOWN;
632     size_t  i;
633     char    *line = NULL;
634     size_t  linecap = 0;

636     sbuf_clear(whatis_proto);
637     if ((fp = fopen(page->filename, "r")) == NULL) {
638         warn("%s", page->filename);
639         return;
640     }
641     while (getline(&line, &linecap, fp) > 0) {
642         /* Skip comments */
643         if (strncmp(line, "\\\"\", 3) == 0)
644             continue;
645         switch (state) {
646             /* Haven't reached the NAME section yet */
647             case STATE_UNKNOWN:
648                 if (name_section_line(line, ".SH"))
649                     state = STATE_MANSTYLE;
650                 else if (name_section_line(line, ".Sh"))

```

```

651         state = STATE_MDOCNAME;
652         continue;
653     /* Inside an old-style .SH NAME section */
654     case STATE_MANSTYLE:
655         if (strncmp(line, ".SH", 3) == 0 ||
656             strncmp(line, ".SS", 3) == 0)
657             break;
658         (void) trim_rhs(line);
659         if (strcmp(line, ".") == 0)
660             continue;
661         if (strncmp(line, ".IX", 3) == 0) {
662             line += 3;
663             line = skip_spaces(line);
664         }
665         process_man_line(line);
666         continue;
667     /* Inside a new-style .Sh NAME section (the .Nm part) */
668     case STATE_MDOCNAME:
669         (void) trim_rhs(line);
670         if (strncmp(line, ".Nm", 3) == 0) {
671             process_mdoc_line(line);
672             continue;
673         } else {
674             if (strcmp(line, ".") == 0)
675                 continue;
676             sbuf_append(whatisc_proto, "- ", 2);
677             state = STATE_MDOCDESC;
678         }
679         /* FALLTHROUGH */
680     /* Inside a new-style .Sh NAME section (after the .Nm-s) */
681     case STATE_MDOCDESC:
682         if (strncmp(line, ".Sh", 3) == 0)
683             break;
684         (void) trim_rhs(line);
685         if (strcmp(line, ".") == 0)
686             continue;
687         process_mdoc_line(line);
688         continue;
689     }
690     break;
691 }
692 (void) fclose(fp);
693 sbuf_strip(whatisc_proto, "\t.-");
694 line = sbuf_content(whatisc_proto);
695 /*
696  * Line now contains the appropriate data, but without the
697  * proper indentation or the section appended to each name.
698  */
699 descr = strstr(line, " - ");
700 if (descr == NULL) {
701     descr = strchr(line, ' ');
702     if (descr == NULL)
703         return;
704     *descr++ = '\0';
705 } else {
706     *descr = '\0';
707     descr += 3;
708 }
709 names = sl_init();
710 collect_names(names, line);
711 sbuf_clear(whatisc_final);
712 if (!sl_find(names, page->name) &&
713     no_page_exists(section_dir, names, page->suffix)) {
714     /*
715      * Add the page name since that's the only
716      * thing that man(1) will find.

```

```

717     /*
718     add_whatisc_name(page->name, page->suffix);
719     */
720     for (i = 0; i < names->sl_cur; i++)
721         add_whatisc_name(names->sl_str[i], page->suffix);
722     sl_free(names, 0);
723     /* Remove last ", " */
724     sbuf_retract(whatisc_final, 2);
725     while (sbuf_length(whatisc_final) < INDENT)
726         sbuf_append(whatisc_final, " ", 1);
727     sbuf_append(whatisc_final, " - ", 3);
728     sbuf_append_str(whatisc_final, skip_spaces(descr));
729     (void) sl_add(whatisc_lines, strdup(sbuf_content(whatisc_final)));
730 }
731
732 /*
733  * Sort pages first by inode number, then by name.
734  */
735 static int
736 pagesort(const void *a, const void *b)
737 {
738     const struct page_info *p1 = *(struct page_info * const *) a;
739     const struct page_info *p2 = *(struct page_info * const *) b;
740
741     if (p1->inode == p2->inode)
742         return (strcmp(p1->name, p2->name));
743
744     return (p1->inode - p2->inode);
745 }
746
747 /*
748  * Process a single man section.
749  */
750 static void
751 process_section(char *section_dir)
752 {
753     struct dirent **entries;
754     int nentries;
755     struct page_info **pages;
756     int npages = 0;
757     int i;
758     ino_t prev_inode = 0;
759
760     /* Scan the man section directory for pages */
761     nentries = scandir(section_dir, &entries, NULL, alphasort);
762
763     /* Collect information about man pages */
764     pages = (struct page_info **)calloc(nentries,
765         sizeof(struct page_info *));
766     for (i = 0; i < nentries; i++) {
767         struct page_info *info = new_page_info(section_dir, entries[i]);
768         if (info != NULL)
769             pages[npages++] = info;
770         free(entries[i]);
771     }
772     free(entries);
773     qsort(pages, npages, sizeof(struct page_info *), pagesort);
774
775     /* Process each unique page */
776     for (i = 0; i < npages; i++) {
777         struct page_info *page = pages[i];
778         if (page->inode != prev_inode) {
779             prev_inode = page->inode;
780             process_page(page, section_dir);
781         }
782         free_page_info(page);

```

```
783     }
784     free(pages);
785 }

787 /*
788  * Return whether the directory entry is a man page section.
789  */
790 static int
791 select_sections(const struct dirent *entry)
792 {
793     const char    *p = &entry->d_name[3];

795     if (strncmp(entry->d_name, "man", 3) != 0)
796         return (0);
797     while (*p != '\0') {
798         if (!isalnum(*p++))
799             return (0);
800     }
801     return (1);
802 }

804 /*
805  * Process a single top-level man directory by finding all the
806  * sub-directories named man* and processing each one in turn.
807  */
808 void
809 mwpath(char *path)
810 {
811     FILE          *fp = NULL;
812     struct dirent **entries;
813     int           nsections;
814     int           i;

816     (void) signal(SIGINT, trap_signal);
817     (void) signal(SIGHUP, trap_signal);
818     (void) signal(SIGQUIT, trap_signal);
819     (void) signal(SIGTERM, trap_signal);

821     whatis_proto = new_sbuf();
822     whatis_final = new_sbuf();

824     nsections = scandir(path, &entries, select_sections, alphasort);
825     if ((fp = open_whatis(path)) == NULL)
826         return;
827     for (i = 0; i < nsections; i++) {
828         char    section_dir[MAXPATHLEN];

830         (void) snprintf(section_dir, MAXPATHLEN, "%s/%s",
831             path, entries[i]->d_name);
832         process_section(section_dir);
833         free(entries[i]);
834     }
835     free(entries);
836     finish_whatis(fp, path);
837 }
```

new/usr/src/cmd/man/man.c

1

```
*****
34083 Sat Jul 19 19:24:28 2014
new/usr/src/cmd/man/man.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
21
22 /*
23  * Copyright (c) 1990, 2010, Oracle and/or its affiliates. All rights reserved.
24  * Copyright 2012, Josef 'Jeff' Sipek <jjeffpc@3lbits.net>. All rights reserved.
25  * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
26  * Copyright 2014 Garrett D'Amore <garrett@damore.org>
27  */
28
29 /*      Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988, 1989  AT&T.  */
30 /*      All rights reserved.                                  */
31
32 /*
33  * University Copyright- Copyright (c) 1982, 1986, 1988
34  * The Regents of the University of California
35  * All Rights Reserved
36  *
37  * University Acknowledgment- Portions of this document are derived from
38  * software developed by the University of California, Berkeley, and its
39  * contributors.
40  */
41
42 /*
43  * Find and display reference manual pages. This version includes makewhatis
44  * functionality as well.
45  */
46
47 #include <sys/param.h>
48 #include <sys/stat.h>
49 #include <sys/termios.h>
50 #include <sys/types.h>
51
52 #include <ctype.h>
53 #include <dirent.h>
54 #include <err.h>
55 #include <errno.h>
56 #include <fcntl.h>
```

new/usr/src/cmd/man/man.c

2

```
57 #include <fnmatch.h>
58 #include <limits.h>
59 #include <locale.h>
60 #include <malloc.h>
61 #include <memory.h>
62 #include <regex.h>
63 #include <stdio.h>
64 #include <stdlib.h>
65 #include <string.h>
66 #include <unistd.h>
67
68 #include "man.h"
69
70
71 /* Mapping of old directories to new directories */
72 static const struct map_entry {
73     char    *old_name;
74     char    *new_name;
75 } map[] = {
76     { "3b",      "3ucb"      },
77     { "3e",      "3elf"      },
78     { "3g",      "3gen"      },
79     { "3k",      "3kstat"    },
80     { "3n",      "3socket"   },
81     { "3r",      "3rt"       },
82     { "3s",      "3c"        },
83     { "3t",      "3thr"      },
84     { "3x",      "3curses"   },
85     { "3xc",     "3xcurses"  },
86     { "3xn",     "3xnet"     },
87     { NULL,     NULL        },
88 };
89
90 struct suffix {
91     char *ds;
92     char *fs;
93 };
94
95 /*
96  * Flags that control behavior of build_manpath()
97  *
98  *   BMP_ISPATH           pathv is a vector constructed from PATH.
99  *                       Perform appropriate path translations for
100  *                       manpath.
101  *   BMP_APPEND_DEFMANDIR Add DEFMANDIR to the end if it hasn't
102  *                       already appeared earlier.
103  *   BMP_FALLBACK_DEFMANDIR Append /usr/share/man only if no other
104  *                       manpath (including derived from PATH)
105  *                       elements are valid.
106  */
107 #define BMP_ISPATH           1
108 #define BMP_APPEND_DEFMANDIR 2
109 #define BMP_FALLBACK_DEFMANDIR 4
110
111 /*
112  * When doing equality comparisons of directories, device and inode
113  * comparisons are done. The secnode and dupnode structures are used
114  * to form a list of lists for this processing.
115  */
116 struct secnode {
117     char    *secp;
118     struct secnode *next;
119 };
120 struct dupnode {
121     dev_t    dev; /* from struct stat st_dev */
122     ino_t    ino; /* from struct stat st_ino */
```

```

123     struct secnode *secl; /* sections already considered */
124     struct dupnode *next;
125 };

127 /*
128  * Map directories that may appear in PATH to the corresponding
129  * man directory.
130  */
131 static struct pathmap {
132     char *bindir;
133     char *mandir;
134     dev_t dev;
135     ino_t ino;
136 } bintoman[] = {
137     { "/sbin",          "/usr/share/man,lm",    0, 0 },
138     { "/usr/sbin",     "/usr/share/man,lm",    0, 0 },
139     { "/usr/ucb",      "/usr/share/man,lb",    0, 0 },
140     { "/usr/bin",      "/usr/share/man,1,lm,ls,lt,lc", 0, 0 },
141     { "/usr/xpg4/bin", "/usr/share/man,1",     0, 0 },
142     { "/usr/xpg6/bin", "/usr/share/man,1",     0, 0 },
143     { NULL,            NULL,                                0, 0 },
144 };

146 struct man_node {
147     char *path; /* mandir path */
148     char **secv; /* submandir suffices */
149     int defsrch; /* hint for man -p */
150     int frompath; /* hint for man -d */
151     struct man_node *next;
152 };

154 static int all = 0;
155 static int apropos = 0;
156 static int debug = 0;
157 static int found = 0;
158 static int list = 0;
159 static int makewhatis = 0;
160 static int printmp = 0;
161 static int sargs = 0;
162 static int psoutput = 0;
163 static int lintout = 0;
164 static int whatis = 0;
165 static int makewhatishere = 0;

167 static char *mansec;
168 static char *pager = NULL;

170 static char *addlocale(char *);
171 static struct man_node *build_manpath(char **, int);
172 static void do_makewhatis(struct man_node *);
173 static char *check_config(char *);
174 static int cmp(const void *, const void *);
175 static int dupcheck(struct man_node *, struct dupnode **);
176 static int format(char *, char *, char *, char *);
177 static void free_dupnode(struct dupnode *);
178 static void free_manp(struct man_node *manp);
179 static void freev(char **);
180 static void fullpaths(struct man_node **);
181 static void get_all_sect(struct man_node *);
182 static int getdirs(char *, char ***, int);
183 static void getpath(struct man_node *, char **);
184 static void getsect(struct man_node *, char **);
185 static void init_bintoman(void);
186 static void lower(char *);
187 static void mandir(char **, char *, char *, int);
188 static int manual(struct man_node *, char *);

```

```

189 static char *map_section(char *, char *);
190 static char *path_to_manpath(char *);
191 static void print_manpath(struct man_node *);
192 static void search_whatis(char *, char *);
193 static int searchdir(char *, char *, char *);
194 static void sortdir(DIR *, char **);
195 static char **split(char *, char *);
196 static void usage_man(void);
197 static void usage_whatapro(void);
198 static void usage_catman(void);
199 static void usage_makewhatis(void);
200 static void whatapro(struct man_node *, char *);

202 static char language[MAXPATHLEN]; /* LC_MESSAGES */
203 static char localedir[MAXPATHLEN]; /* locale specific path component */

205 static char *newsection = NULL;

207 static int manwidth = 0;

209 extern const char *__progname;

211 int
212 main(int argc, char **argv)
213 {
214     int c, i;
215     char **pathv;
216     char *manpath = NULL;
217     static struct man_node *mandirs = NULL;
218     int bmp_flags = 0;
219     int ret = 0;
220     char *opts;
221     char *mwstr;
222     int catman = 0;

224     (void) setlocale(LC_ALL, "");
225     (void) strcpy(language, setlocale(LC_MESSAGES, (char *)NULL));
226     if (strcmp("C", language) != 0)
227         (void) strncpy(localedir, language, MAXPATHLEN);

229 #if !defined(TEXT_DOMAIN)
230 #define TEXT_DOMAIN "SYS_TEST"
231 #endif
232     (void) textdomain(TEXT_DOMAIN);

234     if (strcmp(__progname, "apropos") == 0) {
235         apropos++;
236         opts = "M:ds:";
237     } else if (strcmp(__progname, "whatis") == 0) {
238         apropos++;
239         whatis++;
240         opts = "M:ds:";
241     } else if (strcmp(__progname, "catman") == 0) {
242         catman++;
243         makewhatis++;
244         opts = "P:M:w";
245     } else if (strcmp(__progname, "makewhatis") == 0) {
246         makewhatis++;
247         makewhatishere++;
248         manpath = ".";
249         opts = "";
250     } else {
251         opts = "FM:P:T:adfkpr:tw";
252         if (argc > 1 && strcmp(argv[1], "--") == 0) {
253             pager = "cat";
254             optind++;

```

```

255     }
256 }
257
258 opterr = 0;
259 while ((c = getopt(argc, argv, opts)) != -1) {
260     switch (c) {
261         case 'M': /* Respecify path for man pages */
262             manpath = optarg;
263             break;
264         case 'a':
265             all++;
266             break;
267         case 'd':
268             debug++;
269             break;
270         case 'f':
271             whatis++;
272             /*FALLTHROUGH*/
273         case 'k':
274             apropos++;
275             break;
276         case 'l':
277             list++;
278             all++;
279             break;
280         case 'p':
281             printmp++;
282             break;
283         case 's':
284             mansec = optarg;
285             sargs++;
286             break;
287         case 'r':
288             lintout++;
289             break;
290         case 't':
291             psoutput++;
292             break;
293         case 'T':
294         case 'P':
295         case 'F':
296             /* legacy options, compatibility only and ignored */
297             break;
298         case 'w':
299             makewhatis++;
300             break;
301         case '?':
302         default:
303             if (apropos)
304                 usage_whatapro();
305             else if (catman)
306                 usage_catman();
307             else if (makewhatishere)
308                 usage_makewhatis();
309             else
310                 usage_man();
311     }
312 }
313 argc -= optind;
314 argv += optind;
315
316 if (argc == 0) {
317     if (apropos) {
318         (void) fprintf(stderr, gettext("%s what?\n"),
319             __progname);
320         exit(1);

```

```

321     } else if (!printmp && !makewhatis) {
322         (void) fprintf(stderr,
323             gettext("What manual page do you want?\n"));
324         exit(1);
325     }
326 }
327
328 init_bintoman();
329 if (manpath == NULL && (manpath = getenv("MANPATH")) == NULL) {
330     if ((manpath = getenv("PATH")) != NULL)
331         bmp_flags = BMP_ISPATH | BMP_APPEND_DEFMANDIR;
332     else
333         manpath = DEFMANDIR;
334 }
335 pathv = split(manpath, ':');
336 mandirs = build_manpath(pathv, bmp_flags);
337 freev(pathv);
338 fullpaths(&mandirs);
339
340 if (makewhatis) {
341     do_makewhatis(mandirs);
342     exit(0);
343 }
344
345 if (printmp) {
346     print_manpath(mandirs);
347     exit(0);
348 }
349
350 /* Collect environment information */
351 if (isatty(STDOUT_FILENO) && (mwstr = getenv("MANWIDTH")) != NULL &&
352     *mwstr != '\0') {
353     if (strcasecmp(mwstr, "tty") == 0) {
354         struct winsize ws;
355
356         if (ioctl(0, TIOCGWINSZ, &ws) != 0)
357             warn("TIOCGWINSZ");
358         else
359             manwidth = ws.ws_col;
360     } else {
361         manwidth = (int)strtol(mwstr, (char **)NULL, 10);
362         if (manwidth < 0)
363             manwidth = 0;
364     }
365 }
366 if (manwidth != 0) {
367     DPRINTF("-- Using non-standard page width: %d\n", manwidth);
368 }
369
370 if (pager == NULL) {
371     if ((pager = getenv("PAGER")) == NULL || *pager == '\0')
372         pager = PAGER;
373 }
374 DPRINTF("-- Using pager: %s\n", pager);
375
376 for (i = 0; i < argc; i++) {
377     char *cmd;
378     static struct man_node *mp;
379     char *pv[2];
380
381     /*
382     * If full path to command specified, customize
383     * the manpath accordingly.
384     */
385     if ((cmd = strrchr(argv[i], '/')) != NULL) {
386         *cmd = '\0';

```



```

387         if ((pv[0] = strdup(argv[i])) == NULL)
388             err(1, "strdup");
389         pv[1] = NULL;
390         *cmd = '/';
391         mp = build_manpath(pv,
392             BMP_ISPATH | BMP_FALLBACK_DEFMANDIR);
393     } else {
394         mp = mandirs;
395     }
396
397     if (apropos)
398         whatapro(mp, argv[i]);
399     else
400         ret += manual(mp, argv[i]);
401
402     if (mp != NULL && mp != mandirs) {
403         free(pv[0]);
404         free_manp(mp);
405     }
406 }
407
408 return (ret == 0 ? 0 : 1);
409 }
410
411 /*
412  * This routine builds the manpage structure from MANPATH or PATH,
413  * depending on flags. See BMP_* definitions above for valid
414  * flags.
415  */
416 static struct man_node *
417 build_manpath(char **pathv, int flags)
418 {
419     struct man_node *manpage = NULL;
420     struct man_node *currp = NULL;
421     struct man_node *lastp = NULL;
422     char **p;
423     char *q;
424     char *mand = NULL;
425     char *mandir = DEFMANDIR;
426     int s;
427     struct dupnode *didup = NULL;
428     struct stat sb;
429
430     s = sizeof (struct man_node);
431     for (p = pathv; *p != NULL; ) {
432         if (flags & BMP_ISPATH) {
433             if ((mand = path_to_manpath(*p)) == NULL)
434                 goto next;
435             free(*p);
436             *p = mand;
437         }
438         q = split(*p, ',');
439         if (stat(q[0], &sb) != 0 || (sb.st_mode & S_IFDIR) == 0) {
440             freev(q);
441             goto next;
442         }
443
444         if (access(q[0], R_OK | X_OK) == 0) {
445             /*
446              * Some element exists. Do not append DEFMANDIR as a
447              * fallback.
448              */
449             flags &= ~BMP_FALLBACK_DEFMANDIR;
450
451             if ((currp = (struct man_node *)calloc(1, s)) == NULL)
452                 err(1, "calloc");

```

```

454         currp->frompath = (flags & BMP_ISPATH);
455
456         if (manpage == NULL)
457             lastp = manpage = currp;
458
459         getpath(currp, p);
460         getsect(currp, p);
461
462         /*
463          * If there are no new elements in this path,
464          * do not add it to the manpage list.
465          */
466         if (dupcheck(currp, &didup) != 0) {
467             freev(currp->secv);
468             free(currp);
469         } else {
470             currp->next = NULL;
471             if (currp != manpage)
472                 lastp->next = currp;
473             lastp = currp;
474         }
475     }
476     freev(q);
477 next:
478     /*
479      * Special handling of appending DEFMANDIR. After all pathv
480      * elements have been processed, append DEFMANDIR if needed.
481      */
482     if (p == &mandir)
483         break;
484     p++;
485     if (*p != NULL)
486         continue;
487     if (flags & (BMP_APPEND_DEFMANDIR | BMP_FALLBACK_DEFMANDIR)) {
488         p = &mandir;
489         flags &= ~BMP_ISPATH;
490     }
491 }
492
493 free_dupnode(didup);
494
495 return (manpage);
496 }
497
498 /*
499  * Store the mandir path into the manp structure.
500  */
501 static void
502 getpath(struct man_node *manp, char **pv)
503 {
504     char *s = *pv;
505     int i = 0;
506
507     while (*s != '\0' && *s != ',')
508         i++, s++;
509
510     if ((manp->path = (char *)malloc(i + 1)) == NULL)
511         err(1, "malloc");
512     (void) strncpy(manp->path, *pv, i + 1);
513 }
514
515 /*
516  * Store the mandir's corresponding sections (submandir
517  * directories) into the manp structure.
518  */

```

```

519 static void
520 getsect(struct man_node *manp, char **pv)
521 {
522     char    *sections;
523     char    **sectp;
524
525     /* Just store all sections when doing makewhatis or apropos/whatis */
526     if (makewhatis || apropos) {
527         manp->defsrch = 1;
528         DPRINTF("-- Adding %s\n", manp->path);
529         manp->secv = NULL;
530         get_all_sect(manp);
531     } else if (sargs) {
532         manp->secv = split(mansec, ',');
533         for (sectp = manp->secv; *sectp; sectp++)
534             lower(*sectp);
535     } else if ((sections = strchr(*pv, ',')) != NULL) {
536         DPRINTF("-- Adding %s: MANSECTS=%s\n", manp->path, sections);
537         manp->secv = split(++sections, ',');
538         for (sectp = manp->secv; *sectp; sectp++)
539             lower(*sectp);
540         if (*manp->secv == NULL)
541             get_all_sect(manp);
542     } else if ((sections = check_config(*pv)) != NULL) {
543         manp->defsrch = 1;
544         DPRINTF("-- Adding %s: from %s, MANSECTS=%s\n", manp->path,
545             CONFIG, sections);
546         manp->secv = split(sections, ',');
547         for (sectp = manp->secv; *sectp; sectp++)
548             lower(*sectp);
549         if (*manp->secv == NULL)
550             get_all_sect(manp);
551     } else {
552         manp->defsrch = 1;
553         DPRINTF("-- Adding %s: default sort order\n", manp->path);
554         manp->secv = NULL;
555         get_all_sect(manp);
556     }
557 }
558
559 /*
560 * Get suffices of all sub-mandir directories in a mandir.
561 */
562 static void
563 get_all_sect(struct man_node *manp)
564 {
565     DIR      *dp;
566     char     **dirv;
567     char     **dv;
568     char     **p;
569     char     *prev = NULL;
570     char     *tmp = NULL;
571     int      maxentries = MAXTOKENS;
572     int      entries = 0;
573
574     if ((dp = opendir(manp->path)) == 0)
575         return;
576
577     sortdir(dp, &dirv);
578
579     (void) closedir(dp);
580
581     if (manp->secv == NULL) {
582         if ((manp->secv = malloc(maxentries * sizeof(char *))) == NULL)
583             err(1, "malloc");
584     }

```

```

586         for (dv = dirv, p = manp->secv; *dv; dv++) {
587             if (strcmp(*dv, CONFIG) == 0) {
588                 free(*dv);
589                 continue;
590             }
591
592             free(tmp);
593             if ((tmp = strdup(*dv + 3)) == NULL)
594                 err(1, "strdup");
595
596             if (prev != NULL && strcmp(prev, tmp) == 0) {
597                 free(*dv);
598                 continue;
599             }
600
601             free(prev);
602             if ((prev = strdup(*dv + 3)) == NULL)
603                 err(1, "strdup");
604
605             if ((*p = strdup(*dv + 3)) == NULL)
606                 err(1, "strdup");
607
608             p++; entries++;
609
610             if (entries == maxentries) {
611                 maxentries += MAXTOKENS;
612                 if ((manp->secv = realloc(manp->secv,
613                     sizeof(char *) * maxentries)) == NULL)
614                     err(1, "realloc");
615                 p = manp->secv + entries;
616             }
617             free(*dv);
618         }
619         free(tmp);
620         free(prev);
621         *p = NULL;
622         free(dirv);
623     }
624
625 /*
626 * Build whatis databases.
627 */
628 static void
629 do_makewhatis(struct man_node *manp)
630 {
631     struct man_node *p;
632     char     *ldir;
633
634     for (p = manp; p != NULL; p = p->next) {
635         ldir = addlocale(p->path);
636         if (*localedir != '\0' && getdirs(ldir, NULL, 0) > 0)
637             mwpath(ldir);
638         free(ldir);
639         mwpath(p->path);
640     }
641 }
642
643 /*
644 * Count mandirs under the given manpath
645 */
646 static int
647 getdirs(char *path, char ***dirv, int flag)
648 {
649     DIR      *dp;
650     struct dirent *d;

```

```

651     int             n = 0;
652     int             maxentries = MAXDIRS;
653     char            **dv = NULL;

655     if ((dp = opendir(path)) == NULL)
656         return (0);

658     if (flag) {
659         if ((*dirv = malloc(sizeof (char *) *
660             maxentries)) == NULL)
661             err(1, "malloc");
662         dv = *dirv;
663     }
664     while ((d = readdir(dp)) {
665         if (strncmp(d->d_name, "man", 3) != 0)
666             continue;
667         n++;

669         if (flag) {
670             if ((*dv = strdup(d->d_name + 3)) == NULL)
671                 err(1, "strdup");
672             dv++;
673             if ((dv - *dirv) == maxentries) {
674                 int     entries = maxentries;

676                 maxentries += MAXTOKENS;
677                 if ((*dirv = realloc(*dirv,
678                     sizeof (char *) * maxentries)) == NULL)
679                     err(1, "realloc");
680                 dv = *dirv + entries;
681             }
682         }
683     }

685     (void) closedir(dp);
686     return (n);
687 }

690 /*
691  * Find matching whatis or apropos entries.
692  */
693 static void
694 whatapro(struct man_node *manp, char *word)
695 {
696     char            whatpath[MAXPATHLEN];
697     struct man_node *b;
698     char            *ldir;

700     for (b = manp; b != NULL; b = b->next) {
701         if (*localedir != '\0') {
702             ldir = addlocale(b->path);
703             if (getdirs(ldir, NULL, 0) != 0) {
704                 (void) snprintf(whatpath, sizeof (whatpath),
705                     "%s/%s", ldir, WHATIS);
706                 search_whatis(whatpath, word);
707             }
708             free(ldir);
709         }
710         (void) snprintf(whatpath, sizeof (whatpath), "%s/%s", b->path,
711             WHATIS);
712         search_whatis(whatpath, word);
713     }
714 }

716 static void

```

```

717 search_whatis(char *whatpath, char *word)
718 {
719     FILE            *fp;
720     char            *line = NULL;
721     size_t          linecap = 0;
722     char            *pkwd;
723     regex_t         preg;
724     char            **ss = NULL;
725     char            s[MAXNAMELEN];
726     int             i;

728     if ((fp = fopen(whatpath, "r")) == NULL) {
729         perror(whatpath);
730         return;
731     }

733     DPRPRINTF("-- Found %s: %s\n", WHATIS, whatpath);

735     /* Build keyword regex */
736     if (asprintf(&pkwd, "%s%s%s", (whatis) ? "\\<" : "",
737         word, (whatis) ? "\\>" : "") == -1)
738         err(1, "asprintf");

740     if (regcomp(&preg, pkwd, REG_BASIC | REG_ICASE | REG_NOSUB) != 0)
741         err(1, "regcomp");

743     if (sargs)
744         ss = split(mansec, ',');

746     while (getline(&line, &linecap, fp) > 0) {
747         if (regex(&preg, line, 0, NULL, 0) == 0) {
748             if (sargs) {
749                 /* Section-restricted search */
750                 for (i = 0; ss[i] != NULL; i++) {
751                     (void) snprintf(s, sizeof (s), "(%s)",
752                         ss[i]);
753                     if (strstr(line, s) != NULL) {
754                         (void) printf("%s", line);
755                         break;
756                     }
757                 }
758             } else {
759                 (void) printf("%s", line);
760             }
761         }
762     }

764     if (ss != NULL)
765         freev(ss);
766     free(pkwd);
767     (void) fclose(fp);
768 }

771 /*
772  * Split a string by specified separator.
773  */
774 static char **
775 split(char *s1, char sep)
776 {
777     char            **tokv, **vp;
778     char            *mp = s1, *tp;
779     int             maxentries = MAXTOKENS;
780     int             entries = 0;

782     if ((tokv = vp = malloc(maxentries * sizeof (char *))) == NULL)

```

```

783         err(1, "malloc");
785     for (; mp && *mp; mp = tp) {
786         tp = strchr(mp, sep);
787         if (mp == tp) {
788             tp++;
789             continue;
790         }
791         if (tp) {
792             size_t len;
794             len = tp - mp;
795             if ((*vp = (char *)malloc(sizeof (char) *
796                 len + 1)) == NULL)
797                 err(1, "malloc");
798             (void) strncpy(*vp, mp, len);
799             *(*vp + len) = '\0';
800             tp++;
801             vp++;
802         } else {
803             if ((*vp = strdup(mp)) == NULL)
804                 err(1, "strdup");
805             vp++;
806         }
807         entries++;
808         if (entries == maxentries) {
809             maxentries += MAXTOKENS;
810             if ((tokv = realloc(tokv,
811                 maxentries * sizeof (char *))) == NULL)
812                 err(1, "realloc");
813             vp = tokv + entries;
814         }
815     }
816     *vp = 0;
818     return (tokv);
819 }
821 /*
822  * Free a vector allocated by split()
823  */
824 static void
825 freev(char **v)
826 {
827     int i;
828     if (v != NULL) {
829         for (i = 0; v[i] != NULL; i++) {
830             free(v[i]);
831         }
832         free(v);
833     }
834 }
836 /*
837  * Convert paths to full paths if necessary
838  */
839 static void
840 fullpaths(struct man_node **manp_head)
841 {
842     char *cwd = NULL;
843     char *p;
844     int cwd_gotten = 0;
845     struct man_node *manp = *manp_head;
846     struct man_node *b;
847     struct man_node *prev = NULL;

```

```

849     for (b = manp; b != NULL; b = b->next) {
850         if (*(b->path) == '/') {
851             prev = b;
852             continue;
853         }
855         if (!cwd_gotten) {
856             cwd = getcwd(NULL, MAXPATHLEN);
857             cwd_gotten = 1;
858         }
860         if (cwd) {
861             /* Relative manpath with cwd: make absolute */
862             if (asprintf(&p, "%s/%s", cwd, b->path) == -1)
863                 err(1, "asprintf");
864             free(b->path);
865             b->path = p;
866         } else {
867             /* Relative manpath but no cwd: omit path entry */
868             if (prev)
869                 prev->next = b->next;
870             else
871                 *manp_head = b->next;
873             free_manp(b);
874         }
875     }
876     free(cwd);
877 }
879 /*
880  * Free a man_node structure and its contents
881  */
882 static void
883 free_manp(struct man_node *manp)
884 {
885     char **p;
887     free(manp->path);
888     p = manp->secv;
889     while ((p != NULL) && (*p != NULL)) {
890         free(*p);
891         p++;
892     }
893     free(manp->secv);
894     free(manp);
895 }
898 /*
899  * Map (in place) to lower case.
900  */
901 static void
902 lower(char *s)
903 {
905     if (s == 0)
906         return;
907     while (*s) {
908         if (isupper(*s))
909             *s = tolower(*s);
910         s++;
911     }
912 }

```

```

915 /*
916  * Compare function for qsort().
917  * Sort first by section, then by prefix.
918  */
919 static int
920 cmp(const void *arg1, const void *arg2)
921 {
922     int     n;
923     char    **p1 = (char **)arg1;
924     char    **p2 = (char **)arg2;
925
926     /* By section */
927     if ((n = strcmp(*p1 + 3, *p2 + 3)) != 0)
928         return (n);
929
930     /* By prefix reversed */
931     return (strncmp(*p2, *p1, 3));
932 }
933
934
935 /*
936  * Find a manpage.
937  */
938 static int
939 manual(struct man_node *manp, char *name)
940 {
941     struct man_node *p;
942     struct man_node *local;
943     int     ndirs = 0;
944     char    *ldir;
945     char    *ldirs[2];
946     char    *fullname = name;
947     char    *slash;
948
949     if ((slash = strrchr(name, '/')) != NULL)
950         name = slash + 1;
951
952     /* For each path in MANPATH */
953     found = 0;
954
955     for (p = manp; p != NULL; p = p->next) {
956         DPRINTF("-- Searching mandir: %s\n", p->path);
957
958         if (*locale_dir != '\0') {
959             ldir = addlocale(p->path);
960             ndirs = getdirs(ldir, NULL, 0);
961             if (ndirs != 0) {
962                 ldirs[0] = ldir;
963                 ldirs[1] = NULL;
964                 local = build_manpath(ldirs, 0);
965                 DPRINTF("-- Locale specific subdir: %s\n",
966                     ldir);
967                 mandir(local->secv, ldir, name, 1);
968                 free_manp(local);
969             }
970             free(ldir);
971         }
972
973         /*
974          * Locale mandir not valid, man page in locale
975          * mandir not found, or -a option present
976          */
977         if (ndirs == 0 || !found || !all)
978             mandir(p->secv, p->path, name, 0);
979
980         if (found && !all)

```

```

981             break;
982         }
983
984         if (!found) {
985             if (sargs) {
986                 (void) fprintf(stderr, gettext(
987                     "No manual entry for %s in section(s) %s\n",
988                     fullname, mansec);
989             } else {
990                 (void) fprintf(stderr,
991                     gettext("No manual entry for %s\n"), fullname);
992             }
993         }
994     }
995
996     return (!found);
997 }
998
999
1000 /*
1001  * For a specified manual directory, read, store and sort section subdirs.
1002  * For each section specified, find and search matching subdirs.
1003  */
1004 static void
1005 mandir(char **secv, char *path, char *name, int lspec)
1006 {
1007     DIR     *dp;
1008     char    **dirv;
1009     char    **dv, **pdv;
1010     int     len, dslen;
1011
1012     if ((dp = opendir(path)) == NULL)
1013         return;
1014
1015     if (lspec)
1016         DPRINTF("-- Searching mandir: %s\n", path);
1017
1018     sortdir(dp, &dirv);
1019
1020     /* Search in the order specified by MANSECTS */
1021     for (; *secv; secv++) {
1022         len = strlen(*secv);
1023         for (dv = dirv; *dv; dv++) {
1024             dslen = strlen(*dv + 3);
1025             if (dslen > len)
1026                 len = dslen;
1027             if (**secv == '\\') {
1028                 if (strcmp(*secv + 1, *dv + 3) != 0)
1029                     continue;
1030             } else if (strncasecmp(*secv, *dv + 3, len) != 0) {
1031                 if (!all &&
1032                     (newsection = map_section(*secv, path))
1033                     == NULL) {
1034                     continue;
1035                 }
1036                 if (newsection == NULL)
1037                     newsection = "";
1038                 if (strncmp(newsection, *dv + 3, len) != 0) {
1039                     continue;
1040                 }
1041             }
1042
1043             if (searchdir(path, *dv, name) == 0)
1044                 continue;
1045
1046             if (!all) {

```

```

1047         pdv = dirv;
1048         while (*pdv) {
1049             free(*pdv);
1050             pdv++;
1051         }
1052         (void) closedir(dp);
1053         free(dirv);
1054         return;
1055     }
1056
1057     if (all && **dv == 'm' && *(dv + 1) &&
1058         strcmp(*(dv + 1) + 3, *dv + 3) == 0)
1059         dv++;
1060 }
1061
1062 pdv = dirv;
1063 while (*pdv != NULL) {
1064     free(*pdv);
1065     pdv++;
1066 }
1067 free(dirv);
1068 (void) closedir(dp);
1069 }
1070
1071 /*
1072  * Sort directories.
1073  */
1074 static void
1075 sortdir(DIR *dp, char ***dirv)
1076 {
1077     struct dirent *d;
1078     char **dv;
1079     int maxentries = MAXDIRS;
1080     int entries = 0;
1081
1082     if ((dv = *dirv = malloc(sizeof (char *) *
1083         maxentries)) == NULL)
1084         err(1, "malloc");
1085     dv = *dirv;
1086
1087     while ((d = readdir(dp)) {
1088         if (strcmp(d->d_name, ".") == 0 ||
1089             strcmp(d->d_name, "..") == 0)
1090             continue;
1091
1092         if (strncmp(d->d_name, "man", 3) == 0 ||
1093             strncmp(d->d_name, "cat", 3) == 0) {
1094             if ((*dv = strdup(d->d_name)) == NULL)
1095                 err(1, "strdup");
1096             dv++;
1097             entries++;
1098             if (entries == maxentries) {
1099                 maxentries += MAXDIRS;
1100                 if ((*dirv = realloc(*dirv,
1101                     sizeof (char *) * maxentries)) == NULL)
1102                     err(1, "realloc");
1103                 dv = *dirv + entries;
1104             }
1105         }
1106     }
1107     *dv = 0;
1108
1109     qsort((void *)*dirv, dv - *dirv, sizeof (char *), cmp);
1110 }

```

```

1114 /*
1115  * Search a section subdir for a given manpage.
1116  */
1117 static int
1118 searchdir(char *path, char *dir, char *name)
1119 {
1120     DIR *sdp;
1121     struct dirent *sd;
1122     char sectpath[MAXPATHLEN];
1123     char file[MAXNAMLEN];
1124     char dname[MAXPATHLEN];
1125     char *last;
1126     int nlen;
1127
1128     (void) snprintf(sectpath, sizeof (sectpath), "%s/%s", path, dir);
1129     (void) snprintf(file, sizeof (file), "%s.", name);
1130
1131     if ((sdp = opendir(sectpath)) == NULL)
1132         return (0);
1133
1134     while ((sd = readdir(sdp)) {
1135         char *pname;
1136
1137         if ((pname = strdup(sd->d_name)) == NULL)
1138             err(1, "strdup");
1139         if ((last = strrchr(pname, '.')) != NULL &&
1140             (strcmp(last, ".gz") == 0 || strcmp(last, ".bz2") == 0))
1141             *last = '\0';
1142         last = strrchr(pname, '.');
1143         nlen = last - pname;
1144         (void) snprintf(dname, sizeof (dname), "%.*s.", nlen, pname);
1145         if (strcmp(dname, file) == 0 ||
1146             strcmp(pname, name) == 0) {
1147             (void) format(path, dir, name, sd->d_name);
1148             (void) closedir(sdp);
1149             free(pname);
1150             return (1);
1151         }
1152         free(pname);
1153     }
1154     (void) closedir(sdp);
1155
1156     return (0);
1157 }
1158
1159 /*
1160  * Check the hash table of old directory names to see if there is a
1161  * new directory name.
1162  */
1163 static char *
1164 map_section(char *section, char *path)
1165 {
1166     int i;
1167     char fullpath[MAXPATHLEN];
1168
1169     if (list) /* -l option fall through */
1170         return (NULL);
1171
1172     for (i = 0; map[i].new_name != NULL; i++) {
1173         if (strcmp(section, map[i].old_name) == 0) {
1174             (void) snprintf(fullpath, sizeof (fullpath),
1175                 "%s/man%s", path, map[i].new_name);
1176             if (!access(fullpath, R_OK | X_OK)) {
1177                 return (map[i].new_name);
1178             } else {

```

```

1179         return (NULL);
1180     }
1181 }
1182
1184     return (NULL);
1185 }

```

```

1187 /*
1188  * Format the manpage.
1189  */
1190 static int
1191 format(char *path, char *dir, char *name, char *pg)
1192 {
1193     char      manpname[MAXPATHLEN], catpname[MAXPATHLEN];
1194     char      cmdbuf[BUFSIZ], tmpbuf[BUFSIZ];
1195     char      *cattool;
1196     int       utf8 = 0;
1197     struct stat sbman, sbcat;

```

```

1199     found++;

```

```

1201     if (list) {
1202         (void) printf(gettext("%s(%s)\t-M %s\n"), name, dir + 3, path);
1203         return (-1);
1204     }

```

```

1206     (void) snprintf(manpname, sizeof (manpname), "%s/man%s/%s", path,
1207                    dir + 3, pg);
1208     (void) snprintf(catpname, sizeof (catpname), "%s/cat%s/%s", path,
1209                    dir + 3, pg);

```

```

1211     /* Can't do PS output if manpage doesn't exist */
1212     if (stat(manpname, &sbman) != 0 && (psoutput|lintout))
1213         return (-1);

```

```

1215     /*
1216      * If both manpage and catpage do not exist, manpname is
1217      * broken symlink, most likely.
1218      */
1219     if (stat(catpname, &sbcat) != 0 && stat(manpname, &sbman) != 0)
1220         err(1, "%s", manpname);

```

```

1222     /* Setup cattool */
1223     if (fnmatch("*.gz", manpname, 0) == 0)
1224         cattool = "gzcat";
1225     else if (fnmatch("*.bz2", manpname, 0) == 0)
1226         cattool = "bzcat";
1227     else
1228         cattool = "cat";

```

```

1230     /* Preprocess UTF-8 input with preconv (could be smarter) */
1231     if (strstr(path, "UTF-8") != NULL)
1232         utf8 = 1;

```

```

1234     if (psoutput) {
1235         (void) snprintf(cmdbuf, BUFSIZ,
1236                        "cd %s; %s %s%s | mandoc -Tps | lp -Tpostscript",
1237                        path, cattool, manpname,
1238                        utf8 ? " | " PRECONV " -e UTF-8" : "");
1239         DPRINTF("-- Using manpage: %s\n", manpname);
1240         goto cmd;
1241     } else if (lintout) {
1242         (void) snprintf(cmdbuf, BUFSIZ,
1243                        "cd %s; %s %s%s | mandoc -Tlint",
1244                        path, cattool, manpname,

```

```

1245         utf8 ? " | " PRECONV " -e UTF-8" : "");
1246         DPRINTF("-- Linting manpage: %s\n", manpname);
1247         goto cmd;
1248     }

```

```

1250     /*
1251      * Output catpage if:
1252      * - manpage doesn't exist
1253      * - output width is standard and catpage is recent enough
1254      */
1255     if (stat(manpname, &sbman) != 0 || (manwidth == 0 &&
1256         stat(catpname, &sbcat) == 0 && sbcat.st_mtime >= sbman.st_mtime)) {
1257         DPRINTF("-- Using catpage: %s\n", catpname);
1258         (void) snprintf(cmdbuf, BUFSIZ, "%s %s", pager, catpname);
1259         goto cmd;
1260     }

```

```

1262     DPRINTF("-- Using manpage: %s\n", manpname);
1263     if (manwidth > 0)
1264         (void) snprintf(tmpbuf, BUFSIZ, "-Owidth=%d ", manwidth);
1265     (void) snprintf(cmdbuf, BUFSIZ, "cd %s; %s %s%s | mandoc -T%s %s | %s",
1266                    path, cattool, manpname,
1267                    utf8 ? " | " PRECONV " -e UTF-8" : "",
1268                    utf8 ? "utf8" : "ascii", (manwidth > 0) ? tmpbuf : "", pager);

```

```

1270 cmd:
1271     DPRINTF("-- Command: %s\n", cmdbuf);

```

```

1273     if (!debug)
1274         return (system(cmdbuf) == 0);
1275     else
1276         return (0);
1277 }

```

```

1279 /*
1280  * Add <localedir> to the path.
1281  */
1282 static char *
1283 addlocale(char *path)
1284 {
1285     char      *tmp;

```

```

1287     if (asprintf(&tmp, "%s/%s", path, localedir) == -1)
1288         err(1, "asprintf");

```

```

1290     return (tmp);
1291 }

```

```

1293 /*
1294  * Get the order of sections from man.cf.
1295  */
1296 static char *
1297 check_config(char *path)
1298 {
1299     FILE      *fp;
1300     char      *rc = NULL;
1301     char      *sect;
1302     char      fname[MAXPATHLEN];
1303     char      *line = NULL;
1304     size_t    linecap = 0;

```

```

1306     (void) snprintf(fname, MAXPATHLEN, "%s/%s", path, CONFIG);

```

```

1308     if ((fp = fopen(fname, "r")) == NULL)
1309         return (NULL);

```

```

1311 while (getline(&line, &linecap, fp) > 0) {
1312     if ((rc = strstr(line, "MANSECTS")) != NULL)
1313         break;
1314 }

1316 (void) fclose(fp);

1318 if (rc == NULL || (sect = strchr(line, '=')) == NULL)
1319     return (NULL);
1320 else
1321     return (++sect);
1322 }

1325 /*
1326  * Initialize the bintoman array with appropriate device and inode info.
1327  */
1328 static void
1329 init_bintoman(void)
1330 {
1331     int i;
1332     struct stat sb;

1334     for (i = 0; bintoman[i].bindir != NULL; i++) {
1335         if (stat(bintoman[i].bindir, &sb) == 0) {
1336             bintoman[i].dev = sb.st_dev;
1337             bintoman[i].ino = sb.st_ino;
1338         } else {
1339             bintoman[i].dev = NODEV;
1340         }
1341     }
1342 }

1344 /*
1345  * If a duplicate is found, return 1.
1346  * If a duplicate is not found, add it to the dupnode list and return 0.
1347  */
1348 static int
1349 dupcheck(struct man_node *mnp, struct dupnode **dnp)
1350 {
1351     struct dupnode *curdnp;
1352     struct secnode *cursnp;
1353     struct stat sb;
1354     int i;
1355     int rv = 1;
1356     int dupfound;

1358     /* If the path doesn't exist, treat it as a duplicate */
1359     if (stat(mnp->path, &sb) != 0)
1360         return (1);

1362     /* If no sections were found in the man dir, treat it as duplicate */
1363     if (mnp->secv == NULL)
1364         return (1);

1366     /*
1367      * Find the dupnode structure for the previous time this directory
1368      * was looked at. Device and inode numbers are compared so that
1369      * directories that are reached via different paths (e.g. /usr/man and
1370      * /usr/share/man) are treated as equivalent.
1371      */
1372     for (curdnp = *dnp; curdnp != NULL; curdnp = curdnp->next) {
1373         if (curdnp->dev == sb.st_dev && curdnp->ino == sb.st_ino)
1374             break;
1375     }

```

```

1377     /*
1378      * First time this directory has been seen. Add a new node to the
1379      * head of the list. Since all entries are guaranteed to be unique
1380      * copy all sections to new node.
1381      */
1382     if (curdnp == NULL) {
1383         if ((curdnp = calloc(1, sizeof (struct dupnode))) == NULL)
1384             err(1, "calloc");
1385         for (i = 0; mnp->secv[i] != NULL; i++) {
1386             if ((cursnp = calloc(1, sizeof (struct secnode)))
1387                 == NULL)
1388                 err(1, "calloc");
1389             cursnp->next = curdnp->secl;
1390             curdnp->secl = cursnp;
1391             if ((cursnp->secp = strdup(mnp->secv[i])) == NULL)
1392                 err(1, "strdup");
1393         }
1394         curdnp->dev = sb.st_dev;
1395         curdnp->ino = sb.st_ino;
1396         curdnp->next = *dnp;
1397         *dnp = curdnp;
1398         return (0);
1399     }

1401     /*
1402      * Traverse the section vector in the man_node and the section list
1403      * in dupnode cache to eliminate all duplicates from man_node.
1404      */
1405     for (i = 0; mnp->secv[i] != NULL; i++) {
1406         dupfound = 0;
1407         for (cursnp = curdnp->secl; cursnp != NULL;
1408             cursnp = cursnp->next) {
1409             if (strcmp(mnp->secv[i], cursnp->secp) == 0) {
1410                 dupfound = 1;
1411                 break;
1412             }
1413         }
1414         if (dupfound) {
1415             mnp->secv[i][0] = '\0';
1416             continue;
1417         }

1420     /*
1421      * Update curdnp and set return value to indicate that this
1422      * was not all duplicates.
1423      */
1424     if ((cursnp = calloc(1, sizeof (struct secnode))) == NULL)
1425         err(1, "calloc");
1426     cursnp->next = curdnp->secl;
1427     curdnp->secl = cursnp;
1428     if ((cursnp->secp = strdup(mnp->secv[i])) == NULL)
1429         err(1, "strdup");
1430     rv = 0;
1431 }

1433     return (rv);
1434 }

1436 /*
1437  * Given a bindir, return corresponding mandir.
1438  */
1439 static char *
1440 path_to_manpath(char *bindir)
1441 {
1442     char *mand, *p;

```



```

1443     int         i;
1444     struct stat  sb;

1446     /* First look for known translations for specific bin paths */
1447     if (stat(bindir, &sb) != 0) {
1448         return (NULL);
1449     }
1450     for (i = 0; bintoman[i].bindir != NULL; i++) {
1451         if (sb.st_dev == bintoman[i].dev &&
1452             sb.st_ino == bintoman[i].ino) {
1453             if ((mand = strdup(bintoman[i].mandir)) == NULL)
1454                 err(1, "strdup");
1455             if ((p = strchr(mand, ',')) != NULL)
1456                 *p = '\0';
1457             if (stat(mand, &sb) != 0) {
1458                 free(mand);
1459                 return (NULL);
1460             }
1461             if (p != NULL)
1462                 *p = ',';
1463             return (mand);
1464         }
1465     }

1467     /*
1468     * No specific translation found. Try 'dirname $bindir'/share/man
1469     * and 'dirname $bindir'/man
1470     */
1471     if ((mand = malloc(MAXPATHLEN)) == NULL)
1472         err(1, "malloc");
1473     if (strncpy(mand, bindir, MAXPATHLEN) >= MAXPATHLEN) {
1474         free(mand);
1475         return (NULL);
1476     }

1478     /*
1479     * Advance to end of buffer, strip trailing '/'s then remove last
1480     * directory component.
1481     */
1482     for (p = mand; *p != '\0'; p++)
1483         ;
1484     for (; p > mand && *p == '/'; p--)
1485         ;
1486     for (; p > mand && *p != '/'; p--)
1487         ;
1488     if (p == mand && *p == '.') {
1489         if (realpath(".", mand) == NULL) {
1490             free(mand);
1491             return (NULL);
1492         }
1493         for (; *p != '\0'; p++)
1494             ;
1495     } else {
1496         *p = '\0';
1497     }

1499     if (strlcat(mand, "/share/man", MAXPATHLEN) >= MAXPATHLEN) {
1500         free(mand);
1501         return (NULL);
1502     }

1504     if ((stat(mand, &sb) == 0) && S_ISDIR(sb.st_mode)) {
1505         return (mand);
1506     }

1508     /*

```

```

1509     * Strip the /share/man off and try /man
1510     */
1511     *p = '\0';
1512     if (strlcat(mand, "/man", MAXPATHLEN) >= MAXPATHLEN) {
1513         free(mand);
1514         return (NULL);
1515     }
1516     if ((stat(mand, &sb) == 0) && S_ISDIR(sb.st_mode)) {
1517         return (mand);
1518     }

1520     /*
1521     * No man or share/man directory found
1522     */
1523     free(mand);
1524     return (NULL);
1525 }

1527 /*
1528 * Free a linked list of dupnode structs.
1529 */
1530 void
1531 free_dupnode(struct dupnode *dnp) {
1532     struct dupnode *dnp2;
1533     struct secnode *snp;

1535     while (dnp != NULL) {
1536         dnp2 = dnp;
1537         dnp = dnp->next;
1538         while (dnp2->secl != NULL) {
1539             snp = dnp2->secl;
1540             dnp2->secl = dnp2->secl->next;
1541             free(snp->secp);
1542             free(snp);
1543         }
1544         free(dnp2);
1545     }
1546 }

1548 /*
1549 * Print manp linked list to stdout.
1550 */
1551 void
1552 print_manpath(struct man_node *manp)
1553 {
1554     char    colon[2] = "\0\0";
1555     char    **secp;

1557     for (; manp != NULL; manp = manp->next) {
1558         (void) printf("%s%s", colon, manp->path);
1559         colon[0] = ':';

1561         /*
1562         * If man.cf or a directory scan was used to create section
1563         * list, do not print section list again. If the output of
1564         * man -p is used to set MANPATH, subsequent runs of man
1565         * will re-read man.cf and/or scan man directories as
1566         * required.
1567         */
1568         if (manp->defsrch != 0)
1569             continue;

1571         for (secp = manp->secv; *secp != NULL; secp++) {
1572             /*
1573             * Section deduplication may have eliminated some
1574             * sections from the vector. Avoid displaying this

```

```
1575         * detail which would appear as ",," in output
1576         */
1577         if ((*secp)[0] != '\0')
1578             (void) printf(",%s", *secp);
1579     }
1580     }
1581     (void) printf("\n");
1582 }

1584 static void
1585 usage_man(void)
1586 {
1588     (void) fprintf(stderr, gettext(
1589 "usage: man [-alptw] [-M path] [-s section] name ...\n"
1590 "        man [-M path] [-s section] -k keyword ...\n"
1591 "        man [-M path] [-s section] -f keyword ...\n"));
1593     exit(1);
1594 }

1596 static void
1597 usage_whatapro(void)
1598 {
1600     (void) fprintf(stderr, gettext(
1601 "usage: %s [-M path] [-s section] keyword ...\n"),
1602         whatis ? "whatis" : "apropos");
1604     exit(1);
1605 }

1607 static void
1608 usage_catman(void)
1609 {
1610     (void) fprintf(stderr, gettext(
1611 "usage: catman [-M path] [-w]\n"));
1613     exit(1);
1614 }

1616 static void
1617 usage_makewhatis(void)
1618 {
1619     (void) fprintf(stderr, gettext("usage: makewhatis\n"));
1621     exit(1);
1622 }
```

new/usr/src/cmd/man/man.h

1

```
*****
939 Sat Jul 19 19:24:28 2014
new/usr/src/cmd/man/man.h
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
```

```
1 /*
2  * This file and its contents are supplied under the terms of the
3  * Common Development and Distribution License ("CDDL"), version 1.0.
4  * You may only use this file in accordance with the terms of version
5  * 1.0 of the CDDL.
6  *
7  * A full copy of the text of the CDDL should have accompanied this
8  * source. A copy of the CDDL is also available via the Internet at
9  * http://www.illumos.org/license/CDDL.
10 */

12 /*
13  * Copyright 2012 Nexenta Systems, Inc. All rights reserved.
14  * Copyright 2014 Garrett D'Amore <garrett@damore.org>
15  */

17 /*
18  * Common definitions
19  */

21 #ifndef _MAN_H_
22 #define _MAN_H_

24 #define CONFIG          "man.cf"
25 #define DEFMANDIR      "/usr/share/man"
26 #define INDENT          24
27 #define PAGER          "less -ins"
28 #define WHATIS         "whatis"
29 #define PRECONV        "/usr/lib/mandoc_preconv"

31 #define LINE_ALLOC      4096
32 #define MAXDIRS        128
33 #define MAXTOKENS      64

35 #define DPRINTF         if (debug) \
36                        (void) printf

38 void    mwpath(char *path);

40 #endif /* _MAN_H_ */
```

```

*****
2609 Sat Jul 19 19:24:28 2014
new/usr/src/cmd/man/stringlist.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*
2  * Copyright (c) 1994 Christos Zoulas
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions
7  * are met:
8  * 1. Redistributions of source code must retain the above copyright
9  * notice, this list of conditions and the following disclaimer.
10 * 2. Redistributions in binary form must reproduce the above copyright
11 * notice, this list of conditions and the following disclaimer in the
12 * documentation and/or other materials provided with the distribution.
13 * 4. The name of the author may not be used to endorse or promote products
14 * derived from this software without specific prior written permission.
15 *
16 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS
17 * OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
18 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
19 * ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
20 * DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
21 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
22 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
23 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
24 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
25 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
26 * SUCH DAMAGE.
27 */

29 /*
30  * Copyright 2012 Nexenta Systems, Inc. All rights reserved.
31 */

33 #include <err.h>
34 #include <stdio.h>
35 #include <stdlib.h>
36 #include <string.h>

38 #include "stringlist.h"

40 #define _SL_CHUNKSIZE 20

42 stringlist *
43 sl_init(void)
44 {
45     stringlist    *sl;

47     if ((sl = malloc(sizeof (stringlist))) == NULL)
48         err(1, "malloc");

50     sl->sl_cur = 0;
51     sl->sl_max = _SL_CHUNKSIZE;
52     sl->sl_str = malloc(sl->sl_max * sizeof (char *));
53     if (sl->sl_str == NULL)
54         err(1, "malloc");

56     return (sl);

```

```

57 }

59 int
60 sl_add(stringlist *sl, char *name)
61 {
63     if (sl->sl_cur == sl->sl_max - 1) {
64         sl->sl_max += _SL_CHUNKSIZE;
65         sl->sl_str = realloc(sl->sl_str, sl->sl_max * sizeof (char *));
66         if (sl->sl_str == NULL)
67             return (-1);
68     }
69     sl->sl_str[sl->sl_cur++] = name;

71     return (0);
72 }

75 void
76 sl_free(stringlist *sl, int all)
77 {
78     size_t i;

80     if (sl == NULL)
81         return;
82     if (sl->sl_str) {
83         if (all)
84             for (i = 0; i < sl->sl_cur; i++)
85                 free(sl->sl_str[i]);
86         free(sl->sl_str);
87     }
88     free(sl);
89 }

92 char *
93 sl_find(stringlist *sl, char *name)
94 {
95     size_t i;

97     for (i = 0; i < sl->sl_cur; i++)
98         if (strcmp(sl->sl_str[i], name) == 0)
99             return (sl->sl_str[i]);

101     return (NULL);
102 }

```

new/usr/src/cmd/man/stringlist.h

1

2061 Sat Jul 19 19:24:28 2014

new/usr/src/cmd/man/stringlist.h

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /*
2  * Copyright (c) 1994 Christos Zoulas
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions
7  * are met:
8  * 1. Redistributions of source code must retain the above copyright
9  * notice, this list of conditions and the following disclaimer.
10 * 2. Redistributions in binary form must reproduce the above copyright
11 * notice, this list of conditions and the following disclaimer in the
12 * documentation and/or other materials provided with the distribution.
13 * 3. All advertising materials mentioning features or use of this software
14 * must display the following acknowledgement:
15 * This product includes software developed by Christos Zoulas.
16 * 4. The name of the author may not be used to endorse or promote products
17 * derived from this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS
20 * OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
21 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
22 * ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
23 * DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
25 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
26 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
27 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
28 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
29 * SUCH DAMAGE.
30 */
31
32 /*
33  * Copyright 2012 Nexenta Systems, Inc. All rights reserved.
34  */
35
36 #ifndef _STRINGLIST_H
37 #define _STRINGLIST_H
38
39 #include <sys/types.h>
40
41 typedef struct _stringlist {
42     char    **sl_str;
43     size_t  sl_max;
44     size_t  sl_cur;
45 } stringlist;
46
47 stringlist *sl_init(void);
48 int         sl_add(stringlist *, char *);
49 void        sl_free(stringlist *, int);
50 char        *sl_find(stringlist *, char *);
51
52 #endif /* _STRINGLIST_H */
```

new/usr/src/cmd/mandoc/Makefile

1

1245 Sat Jul 19 19:24:28 2014

new/usr/src/cmd/mandoc/Makefile

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2014 Nexenta Systems, Inc. All rights reserved.
14 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
15 #
16 #
17 PROGS=          mandoc mandoc_preconv
18 #
19 # We place preconv in /usr/lib. This is done to avoid conflicting with
20 # GNU groff, which puts it into /usr/bin. We also rename it so that it
21 # will only be seen by mandoc -- it isn't intended for general end-user use.
22 #
23 ROOTPROGS =     $(ROOTBIN)/mandoc $(ROOTLIB)/mandoc_preconv
24 #
25 OBJS=           $(preconv_OBJS) $(mandoc_OBJS)
26 #
27 include         $(SRC)/cmd/Makefile.cmd
28 include         $(SRC)/cmd/mandoc/Makefile.common
29 #
30 .KEEP_STATE:
31 #
32 all:            $(PROGS)
33 #
34 mandoc_preconv: $(preconv_OBJS)
35                 $(LINK.c) $(preconv_OBJS) -o $@ $(LDLIBS)
36                 $(POST_PROCESS)
37 #
38 mandoc:         $(mandoc_OBJS)
39                 $(LINK.c) $(mandoc_OBJS) -o $@ $(LDLIBS)
40                 $(POST_PROCESS)
41 #
42 clean:
43                 $(RM) $(OBJS)
44 #
45 install:        all $(ROOTPROGS)
46 #
47 include         $(SRC)/cmd/Makefile.targ
```

new/usr/src/cmd/mandoc/Makefile.common

1

1121 Sat Jul 19 19:24:28 2014

new/usr/src/cmd/mandoc/Makefile.common

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #
```

```
12 #
13 # Copyright 2012 Nexenta Systems, Inc. All rights reserved.
14 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
15 #
```

```
17 PROGS=      mandoc mandoc_preconv
18 mandoc_OBJS = arch.o att.o chars.o eqn.o eqn_html.o eqn_term.o \
19              html.o lib.o main.o man.o man_hash.o man_html.o \
20              man_macro.o man_term.o man_validate.o mandoc.o mdoc.o \
21              mdoc_argv.o mdoc_hash.o mdoc_html.o mdoc_macro.o \
22              mdoc_man.o mdoc_term.o mdoc_validate.o msec.o out.o \
23              read.o roff.o st.o tbl.o tbl_data.o tbl_html.o \
24              tbl_layout.o tbl_opts.o tbl_term.o term.o term_ascii.o \
25              term_ps.o tree.o vol.o
```

```
27 preconv_OBJS = preconv.o
```

```
29 CFLAGS +=   $(CC_VERBOSE)
```

```
31 CPPFLAGS += -DHAVE_CONFIG_H -DUSE_WCHAR \
32             -DOSNAME="\illumos\" \
33             -DVERSION="\1.12.1\""
```

new/usr/src/cmd/mandoc/THIRDPARTYLICENSE

1

825 Sat Jul 19 19:24:28 2014

new/usr/src/cmd/mandoc/THIRDPARTYLICENSE

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

1 Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>

2 Copyright (c) 2011 Ingo Schwarze <schwarze@openbsd.org>

4 Permission to use, copy, modify, and distribute this software for any
5 purpose with or without fee is hereby granted, provided that the above
6 copyright notice and this permission notice appear in all copies.

8 THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHORS DISCLAIM ALL WARRANTIES
9 WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
10 MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR
11 ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
12 WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
13 ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
14 OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

new/usr/src/cmd/mandoc/THIRDPARTYLICENSE.descrip

1

41 Sat Jul 19 19:24:28 2014

new/usr/src/cmd/mandoc/THIRDPARTYLICENSE.descrip

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

1 MANDOC - FORMAT AND DISPLAY UNIX MANUALS

new/usr/src/cmd/mandoc/arch.c

1

1159 Sat Jul 19 19:24:28 2014

new/usr/src/cmd/mandoc/arch.c

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /*      $Id: arch.c,v 1.9 2011/03/22 14:33:05 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <stdlib.h>
22 #include <string.h>
23 #include <time.h>
24
25 #include "mdoc.h"
26 #include "mandoc.h"
27 #include "libmdoc.h"
28
29 #define LINE(x, y) \
30     if (0 == strcmp(p, x)) return(y);
31
32 const char *
33 mdoc_a2arch(const char *p)
34 {
35
36     #include "arch.in"
37
38     return(NULL);
39 }
```

3281 Sat Jul 19 19:24:28 2014

new/usr/src/cmd/mandoc/arch.in

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 /*      $Id: arch.in,v 1.12 2012/01/28 14:02:17 joerg Exp $ */
2 /*
3  * Copyright (c) 2009 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17
18 /*
19 * This file defines the architecture token of the .Dt prologue macro.
20 * All architectures that your system supports (or the manuals of your
21 * system) should be included here. The right-hand-side is the
22 * formatted output.
23 *
24 * Be sure to escape strings.
25 *
26 * REMEMBER TO ADD NEW ARCHITECTURES TO MDOC.7!
27 */
28
29 LINE("acorn26",      "Acorn26")
30 LINE("acorn32",      "Acorn32")
31 LINE("algor",        "Algor")
32 LINE("alpha",        "Alpha")
33 LINE("amd64",        "AMD64")
34 LINE("amiga",        "Amiga")
35 LINE("amigappc",     "AmigaPPC")
36 LINE("arc",          "ARC")
37 LINE("arm",          "ARM")
38 LINE("arm26",        "ARM26")
39 LINE("arm32",        "ARM32")
40 LINE("armish",       "ARMISH")
41 LINE("aviion",       "AVIION")
42 LINE("atari",        "ATARI")
43 LINE("beagle",       "Beagle")
44 LINE("bebox",        "BeBox")
45 LINE("cats",         "cats")
46 LINE("cesfic",       "CESFIC")
47 LINE("cobalt",       "Cobalt")
48 LINE("dreamcast",   "Dreamcast")
49 LINE("emips",        "EMIPS")
50 LINE("evbarm",       "evbARM")
51 LINE("evbmips",      "evbMIPS")
52 LINE("evbppc",       "evbPPC")
53 LINE("evbsh3",       "evbSH3")
54 LINE("ews4800mips", "EWS4800MIPS")
55 LINE("hp300",        "HP300")
56 LINE("hp700",        "HP700")

```

```

57 LINE("hpcarm",      "HPCARM")
58 LINE("hpcmips",     "HPCMIPS")
59 LINE("hpcsh",       "HPCSH")
60 LINE("hppa",        "HPPA")
61 LINE("hppa64",      "HPPA64")
62 LINE("ia64",        "ia64")
63 LINE("i386",        "i386")
64 LINE("ibmwns",      "IBMwns")
65 LINE("iyonix",      "Iyonix")
66 LINE("landisk",     "LANDISK")
67 LINE("loongson",    "Loongson")
68 LINE("luna68k",     "Luna68k")
69 LINE("luna88k",     "Luna88k")
70 LINE("m68k",        "m68k")
71 LINE("mac68k",      "Mac68k")
72 LINE("macppc",      "MacPPC")
73 LINE("mips",        "MIPS")
74 LINE("mips64",      "MIPS64")
75 LINE("mipsco",      "MIPSCO")
76 LINE("mmeye",       "mmEye")
77 LINE("mvme68k",    "MVME68k")
78 LINE("mvme88k",    "MVME88k")
79 LINE("mvmeppc",    "MVMEPPC")
80 LINE("netwinder",   "NetWinder")
81 LINE("news68k",     "News68k")
82 LINE("news68k",    "NewsMIPS")
83 LINE("next68k",    "NeXT68k")
84 LINE("ofppc",      "OFPPC")
85 LINE("palm",        "Palm")
86 LINE("pc532",       "PC532")
87 LINE("playstation2", "PlayStation2")
88 LINE("pmax",        "PMAX")
89 LINE("pmppc",       "pmPPC")
90 LINE("powerpc",     "PowerPC")
91 LINE("prep",        "PreP")
92 LINE("rs6000",      "RS6000")
93 LINE("sandpoint",   "Sandpoint")
94 LINE("sbmips",      "SBMIPS")
95 LINE("sgi",         "SGI")
96 LINE("sgimips",    "SGIMIPS")
97 LINE("sh3",         "SH3")
98 LINE("shark",       "Shark")
99 LINE("socppc",      "SOCPPC")
100 LINE("solbourne",  "Solbourne")
101 LINE("sparc",       "SPARC")
102 LINE("sparc64",    "SPARC64")
103 LINE("sun2",        "Sun2")
104 LINE("sun3",        "Sun3")
105 LINE("tahoe",       "Tahoe")
106 LINE("vax",         "VAX")
107 LINE("x68k",        "X68k")
108 LINE("x86",         "x86")
109 LINE("x86_64",     "x86_64")
110 LINE("xen",         "Xen")
111 LINE("zaurus",     "Zaurus")

```

new/usr/src/cmd/mandoc/att.c

1

1156 Sat Jul 19 19:24:29 2014

new/usr/src/cmd/mandoc/att.c

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /*      $Id: att.c,v 1.9 2011/03/22 14:33:05 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <stdlib.h>
22 #include <string.h>
23 #include <time.h>
24
25 #include "mdoc.h"
26 #include "mandoc.h"
27 #include "libmdoc.h"
28
29 #define LINE(x, y) \
30     if (0 == strcmp(p, x)) return(y);
31
32 const char *
33 mdoc_a2att(const char *p)
34 {
35
36 #include "att.in"
37
38     return(NULL);
39 }
```

new/usr/src/cmd/mandoc/att.in

1

1738 Sat Jul 19 19:24:29 2014

new/usr/src/cmd/mandoc/att.in

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /*      $Id: att.in,v 1.8 2011/07/31 17:30:33 schwarze Exp $ */
2 /*
3  * Copyright (c) 2009 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17
18 /*
19 * This file defines the AT&T versions of the .At macro. This probably
20 * isn't going to change. The right-hand side is the formatted string.
21 *
22 * Be sure to escape strings.
23 * The non-breaking blanks prevent ending an output line right before
24 * a number. Groff prevent line breaks at the same places.
25 */
26
27 LINE("v1",      "Version\\~1 AT&T UNIX")
28 LINE("v2",      "Version\\~2 AT&T UNIX")
29 LINE("v3",      "Version\\~3 AT&T UNIX")
30 LINE("v4",      "Version\\~4 AT&T UNIX")
31 LINE("v5",      "Version\\~5 AT&T UNIX")
32 LINE("v6",      "Version\\~6 AT&T UNIX")
33 LINE("v7",      "Version\\~7 AT&T UNIX")
34 LINE("32v",     "Version\\~32V AT&T UNIX")
35 LINE("III",     "AT&T System\\~III UNIX")
36 LINE("V",       "AT&T System\\~V UNIX")
37 LINE("V.1",     "AT&T System\\~V Release\\~1 UNIX")
38 LINE("V.2",     "AT&T System\\~V Release\\~2 UNIX")
39 LINE("V.3",     "AT&T System\\~V Release\\~3 UNIX")
40 LINE("V.4",     "AT&T System\\~V Release\\~4 UNIX")
```

```
*****
```

```
3566 Sat Jul 19 19:24:29 2014
new/usr/src/cmd/mandoc/chars.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
```

```
*****
```

```
1 /* $Id: chars.c,v 1.52 2011/11/08 00:15:23 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <assert.h>
23 #include <ctype.h>
24 #include <stdlib.h>
25 #include <string.h>
26
27 #include "mandoc.h"
28 #include "libmandoc.h"
29
30 #define PRINT_HI 126
31 #define PRINT_LO 32
32
33 struct ln {
34     struct ln *next;
35     const char *code;
36     const char *ascii;
37     int unicode;
38 };
39
40 #define LINES_MAX 328
41
42 #define CHAR(in, ch, code) \
43     { NULL, (in), (ch), (code) },
44
45 #define CHAR_TBL_START static struct ln lines[LINES_MAX] = {
46 #define CHAR_TBL_END };
47
48 #include "chars.in"
49
50 struct mchars {
51     struct ln **htab;
52 };
53
54 static const struct ln *find(const struct mchars *,
55     const char *, size_t);
```

```
57 void
58 mchars_free(struct mchars *arg)
59 {
60
61     free(arg->htab);
62     free(arg);
63 }
64
65 struct mchars *
66 mchars_alloc(void)
67 {
68     struct mchars *tab;
69     struct ln **htab;
70     struct ln *pp;
71     int i, hash;
72
73     /*
74      * Constructs a very basic chaining hashtable. The hash routine
75      * is simply the integral value of the first character.
76      * Subsequent entries are chained in the order they're processed.
77      */
78
79     tab = mandoc_malloc(sizeof(struct mchars));
80     htab = mandoc_calloc(PRINT_HI - PRINT_LO + 1, sizeof(struct ln **));
81
82     for (i = 0; i < LINES_MAX; i++) {
83         hash = (int)lines[i].code[0] - PRINT_LO;
84
85         if (NULL == (pp = htab[hash])) {
86             htab[hash] = &lines[i];
87             continue;
88         }
89
90         for ( ; pp->next; pp = pp->next)
91             /* Scan ahead. */ ;
92         pp->next = &lines[i];
93     }
94
95     tab->htab = htab;
96     return(tab);
97 }
98
99 int
100 mchars_spec2cp(const struct mchars *arg, const char *p, size_t sz)
101 {
102     const struct ln *ln;
103
104     ln = find(arg, p, sz);
105     if (NULL == ln)
106         return(-1);
107     return(ln->unicode);
108 }
109
110 char
111 mchars_num2char(const char *p, size_t sz)
112 {
113     int i;
114
115     if ((i = mandoc_strntoi(p, sz, 10)) < 0)
116         return('\0');
117     return(i > 0 && i < 256 && isprint(i) ?
118         /* LINTED */ i : '\0');
119 }
120
121 int
122 mchars_num2uc(const char *p, size_t sz)
```

```
123 {
124     int            i;

126     if ((i = mandoc_strntoi(p, sz, 16)) < 0)
127         return('\0');
128     /* FIXME: make sure we're not in a bogus range. */
129     return(i > 0x80 && i <= 0x10FFFF ? i : '\0');
130 }

132 const char *
133 mchars_spec2str(const struct mchars *arg,
134               const char *p, size_t sz, size_t *rsz)
135 {
136     const struct ln *ln;

138     ln = find(arg, p, sz);
139     if (NULL == ln) {
140         *rsz = 1;
141         return(NULL);
142     }

144     *rsz = strlen(ln->ascii);
145     return(ln->ascii);
146 }

148 static const struct ln *
149 find(const struct mchars *tab, const char *p, size_t sz)
150 {
151     const struct ln *pp;
152     int             hash;

154     assert(p);

156     if (0 == sz || p[0] < PRINT_LO || p[0] > PRINT_HI)
157         return(NULL);

159     hash = (int)p[0] - PRINT_LO;

161     for (pp = tab->htab[hash]; pp; pp = pp->next)
162         if (0 == strncmp(pp->code, p, sz) &&
163             '\0' == pp->code[(int)sz])
164             return(pp);

166     return(NULL);
167 }
```

```

*****
10032 Sat Jul 19 19:24:29 2014
new/usr/src/cmd/mandoc/chars.in
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*
2  $Id: chars.in,v 1.42 2011/10/02 10:02:26 kristaps Exp $ */
3
4 * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
5
6 * Permission to use, copy, modify, and distribute this software for any
7 * purpose with or without fee is hereby granted, provided that the above
8 * copyright notice and this permission notice appear in all copies.
9
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17
18 /*
19 * The ASCII translation tables.
20 *
21 * The left-hand side corresponds to the input sequence (\x, \{xx, \{*(xx
22 * and so on) whose length is listed second element. The right-hand
23 * side is what's produced by the front-end, with the fourth element
24 * being its length.
25 *
26 * XXX - C-escape strings!
27 * XXX - update LINES_MAX if adding more!
28 */
29
30 /* Non-breaking, non-collapsing space uses unit separator. */
31 static const char ascii_nbrsp[2] = { ASCII_NBRSP, '\0' };
32
33 CHAR_TBL_START
34
35 /* Spacing. */
36 CHAR("c",          "",          0)
37 CHAR("0",          " ",         8194)
38 CHAR(" ",         ascii_nbrsp, 160)
39 CHAR("~",         ascii_nbrsp, 160)
40 CHAR("%",         "",          0)
41 CHAR("&",         "",          0)
42 CHAR("^",         "",          0)
43 CHAR("|",         "",          0)
44 CHAR(")",         "",          0)
45
46 /* Accents. */
47 CHAR("a",         "\'",        779)
48 CHAR("a-",       "\' ",       175)
49 CHAR("a.",       "\' .",      729)
50 CHAR("a^",       "\' ^",      770)
51 CHAR("\'",       "\' ",       769)
52 CHAR("aa",       "\' ",       769)
53 CHAR("ga",       "\' ",       768)
54 CHAR("\'",       "\' ",       768)
55 CHAR("ab",       "\' ",       774)
56 CHAR("ac",       "\' ",       807)

```

```

57 CHAR("ad",       "\' ",       776)
58 CHAR("ah",       "\' ",       711)
59 CHAR("ao",       "\' ",       730)
60 CHAR("a-",       "\' ",       771)
61 CHAR("ho",       "\' ",       808)
62 CHAR("ha",       "\' ",       94)
63 CHAR("ti",       "\' ",       126)
64
65 /* Quotes. */
66 CHAR("Bq",       "\' ",       8222)
67 CHAR("Bq",       "\' ",       8218)
68 CHAR("lq",       "\' ",       8220)
69 CHAR("rq",       "\' \'",     8221)
70 CHAR("oq",       "\' ",       8216)
71 CHAR("cq",       "\' ",       8217)
72 CHAR("aq",       "\' ",       39)
73 CHAR("dq",       "\' ",       34)
74 CHAR("Fo",       "\' <",     171)
75 CHAR("Fc",       "\' >",     187)
76 CHAR("fo",       "\' <",     8249)
77 CHAR("fi",       "\' >",     8250)
78
79 /* Brackets. */
80 CHAR("lB",       "[",        91)
81 CHAR("rB",       "]",        93)
82 CHAR("lC",       "{",        123)
83 CHAR("rC",       "}",        125)
84 CHAR("la",       "<",        60)
85 CHAR("ra",       ">",        62)
86 CHAR("bv",       "\' ",       9130)
87 CHAR("braceex", "\' ",       9130)
88 CHAR("bracketlefttp", "\' ", 9121)
89 CHAR("bracketleftbp", "\' ", 9123)
90 CHAR("bracketleftex", "\' ", 9122)
91 CHAR("bracketrighttp", "\' ", 9124)
92 CHAR("bracketrightbp", "\' ", 9126)
93 CHAR("bracketrightex", "\' ", 9125)
94 CHAR("lt",      "\' -",     9127)
95 CHAR("bracelefttp", "\' -", 9127)
96 CHAR("lk",      "\' {",     9128)
97 CHAR("braceleftmid", "\' {", 9128)
98 CHAR("lb",      "\' -",     9129)
99 CHAR("braceleftbp", "\' -", 9129)
100 CHAR("braceleftex", "\' |", 9130)
101 CHAR("rt",      "\' -",     9131)
102 CHAR("bracerighttp", "\' -", 9131)
103 CHAR("rk",      "\' }",     9132)
104 CHAR("bracerightmid", "\' }", 9132)
105 CHAR("rb",      "\' -",     9133)
106 CHAR("bracerightbp", "\' -", 9133)
107 CHAR("bracerightex", "\' |", 9130)
108 CHAR("parenlefttp", "\' /", 9115)
109 CHAR("parenleftbp", "\' \'", 9117)
110 CHAR("parenleftex", "\' |", 9116)
111 CHAR("parenrighttp", "\' \'", 9118)
112 CHAR("parenrightbp", "\' /", 9120)
113 CHAR("parenrightex", "\' |", 9119)
114
115 /* Greek characters. */
116 CHAR("*A",       "A",        913)
117 CHAR("*B",       "B",        914)
118 CHAR("*G",       "G",        915)
119 CHAR("*D",       "D",        916)
120 CHAR("*E",       "E",        917)
121 CHAR("+Z",       "Z",        918)
122 CHAR("*Y",       "H",        919)

```



```

123 CHAR(" *H", "O", 920)
124 CHAR(" *I", "I", 921)
125 CHAR(" *K", "K", 922)
126 CHAR(" *L", "L", 923)
127 CHAR(" *M", "M", 924)
128 CHAR(" *N", "N", 925)
129 CHAR(" *C", "H", 926)
130 CHAR(" *O", "O", 927)
131 CHAR(" *P", "T", 928)
132 CHAR(" *R", "P", 929)
133 CHAR(" *S", ">", 931)
134 CHAR(" *T", "T", 932)
135 CHAR(" *U", "Y", 933)
136 CHAR(" *F", "O", 934)
137 CHAR(" *X", "X", 935)
138 CHAR(" *Q", "Y", 936)
139 CHAR(" *W", "O", 937)
140 CHAR(" *a", "a", 945)
141 CHAR(" *b", "B", 946)
142 CHAR(" *g", "y", 947)
143 CHAR(" *d", "d", 948)
144 CHAR(" *e", "e", 949)
145 CHAR(" *z", "C", 950)
146 CHAR(" *y", "n", 951)
147 CHAR(" *h", "O", 952)
148 CHAR(" *i", "i", 953)
149 CHAR(" *k", "k", 954)
150 CHAR(" *l", "\\", 955)
151 CHAR(" *m", "u", 956)
152 CHAR(" *n", "v", 957)
153 CHAR(" *c", "E", 958)
154 CHAR(" *o", "O", 959)
155 CHAR(" *p", "n", 960)
156 CHAR(" *r", "p", 961)
157 CHAR(" *s", "O", 963)
158 CHAR(" *t", "t", 964)
159 CHAR(" *u", "u", 965)
160 CHAR(" *f", "O", 981)
161 CHAR(" *x", "x", 967)
162 CHAR(" *q", "u", 968)
163 CHAR(" *w", "w", 969)
164 CHAR(" *h", "O", 977)
165 CHAR(" +f", "O", 966)
166 CHAR(" +p", "w", 982)
167 CHAR(" +e", "e", 1013)
168 CHAR(" ts", "s", 962)

170 /* Accented letters. */
171 CHAR(" ,C", "C", 199)
172 CHAR(" ,c", "c", 231)
173 CHAR(" /L", "L", 321)
174 CHAR(" /O", "O", 216)
175 CHAR(" /l", "l", 322)
176 CHAR(" /o", "o", 248)
177 CHAR(" oA", "A", 197)
178 CHAR(" oa", "a", 229)
179 CHAR(" :A", "A", 196)
180 CHAR(" :E", "E", 203)
181 CHAR(" :I", "I", 207)
182 CHAR(" :O", "O", 214)
183 CHAR(" :U", "U", 220)
184 CHAR(" :a", "a", 228)
185 CHAR(" :e", "e", 235)
186 CHAR(" :i", "i", 239)
187 CHAR(" :o", "o", 246)
188 CHAR(" :u", "u", 252)

```

```

189 CHAR(" :y", "y", 255)
190 CHAR(" ^A", "A", 193)
191 CHAR(" ^E", "E", 201)
192 CHAR(" ^I", "I", 205)
193 CHAR(" ^O", "O", 211)
194 CHAR(" ^U", "U", 218)
195 CHAR(" 'a", "a", 225)
196 CHAR(" 'e", "e", 233)
197 CHAR(" 'i", "i", 237)
198 CHAR(" 'o", "o", 243)
199 CHAR(" 'u", "u", 250)
200 CHAR(" ^A", "A", 194)
201 CHAR(" ^E", "E", 202)
202 CHAR(" ^I", "I", 206)
203 CHAR(" ^O", "O", 212)
204 CHAR(" ^U", "U", 219)
205 CHAR(" a", "a", 226)
206 CHAR(" ^e", "e", 234)
207 CHAR(" ^i", "i", 238)
208 CHAR(" ^o", "o", 244)
209 CHAR(" ^u", "u", 251)
210 CHAR(" A", "A", 192)
211 CHAR(" E", "E", 200)
212 CHAR(" I", "I", 204)
213 CHAR(" O", "O", 210)
214 CHAR(" U", "U", 217)
215 CHAR(" a", "a", 224)
216 CHAR(" e", "e", 232)
217 CHAR(" i", "i", 236)
218 CHAR(" o", "o", 242)
219 CHAR(" u", "u", 249)
220 CHAR(" -A", "A", 195)
221 CHAR(" -N", "N", 209)
222 CHAR(" -O", "O", 213)
223 CHAR(" -a", "a", 227)
224 CHAR(" -n", "n", 241)
225 CHAR(" -o", "o", 245)

227 /* Arrows and lines. */
228 CHAR("<-", "<-", 8592)
229 CHAR(">", ">", 8594)
230 CHAR("<>", "<>", 8596)
231 CHAR("da", "v", 8595)
232 CHAR("ua", "A", 8593)
233 CHAR("va", "v", 8597)
234 CHAR("lA", "<=", 8656)
235 CHAR("rA", "=>", 8658)
236 CHAR("hA", "<=>", 8660)
237 CHAR("dA", "v", 8659)
238 CHAR("uA", "A", 8657)
239 CHAR("vA", "A^v", 8661)

241 /* Logic. */
242 CHAR("AN", "A", 8743)
243 CHAR("OR", "v", 8744)
244 CHAR("no", "-", 172)
245 CHAR("tno", "~", 172)
246 CHAR("te", "3", 8707)
247 CHAR("fa", "v", 8704)
248 CHAR("st", ")", 8715)
249 CHAR("tf", ". : .", 8756)
250 CHAR("3d", ". : .", 8756)
251 CHAR("or", "|", 124)

253 /* Mathematical. */
254 CHAR("pl", "+", 43)

```

```

255 CHAR("mi",      "-_",      8722)
256 CHAR("-",       "-_",      45)
257 CHAR("-+",      "-+_",      8723)
258 CHAR("+-",      "+-_",      177)
259 CHAR("t+-",     "t+-_",   177)
260 CHAR("pc",      "._",      183)
261 CHAR("md",      "._",      8901)
262 CHAR("mu",      "x_",      215)
263 CHAR("tmu",     "x_",      215)
264 CHAR("c*",      "x_",      8855)
265 CHAR("c+",      "x_",      8853)
266 CHAR("di",      "-:~",     247)
267 CHAR("tdi",     "-:~",     247)
268 CHAR("f/",      "/_",      8260)
269 CHAR("f**",     "**_",      8727)
270 CHAR("f<=",    "<=_",    8804)
271 CHAR("f>=",    ">=_",    8805)
272 CHAR("f<<=",  "<<=_",   8810)
273 CHAR("f>>=",  ">>=_",   8811)
274 CHAR("eq",      "=_",      61)
275 CHAR("i=",      "i=_",    8800)
276 CHAR("e=",      "e=_",    8801)
277 CHAR("ne",      "ne=",    8802)
278 CHAR("e~",      "e~=",    8773)
279 CHAR("e--",     "e--",    8771)
280 CHAR("ap",      "ap",      8764)
281 CHAR("a~",      "a~",      8776)
282 CHAR("a=",      "a=",      8780)
283 CHAR("pt",      "oc",      8733)
284 CHAR("es",      "{ }",     8709)
285 CHAR("mo",      "E",      8712)
286 CHAR("nm",      "IE",     8713)
287 CHAR("sb",      "(=",     8834)
288 CHAR("nb",      "( !=",   8836)
289 CHAR("sp",      "=)",     8835)
290 CHAR("nc",      "!=",     8837)
291 CHAR("ib",      "(=",     8838)
292 CHAR("ip",      "(=",     8839)
293 CHAR("ca",      "(^)",    8745)
294 CHAR("cu",      "U",      8746)
295 CHAR("/_ ",     "/_ ",    8736)
296 CHAR("pp",      "_|_",    8869)
297 CHAR("is",      "I",      8747)
298 CHAR("integral", "I",      8747)
299 CHAR("sum",     "E",      8721)
300 CHAR("product", "T",      8719)
301 CHAR("coproduct", "U",      8720)
302 CHAR("gr",      "v",      8711)
303 CHAR("sr",      "\\/",    8730)
304 CHAR("sqrt",   "\\/",    8730)
305 CHAR("lc",      "|~",     8968)
306 CHAR("rc",      "~|",     8969)
307 CHAR("lf",      "|_",     8970)
308 CHAR("rf",      "_|",     8971)
309 CHAR("if",      "oo",     8734)
310 CHAR("Ah",      "N",      8501)
311 CHAR("Im",      "I",      8465)
312 CHAR("Re",      "R",      8476)
313 CHAR("pd",      "a",      8706)
314 CHAR("-h",     "/h",     8463)
315 CHAR("12",     "1/2",    189)
316 CHAR("14",     "1/4",    188)
317 CHAR("34",     "3/4",    190)

319 /* Ligatures. */
320 CHAR("ff",      "ff",      64256)

```

```

321 CHAR("fi",      "fi",      64257)
322 CHAR("fl",      "fl",      64258)
323 CHAR("Fi",      "ffi",    64259)
324 CHAR("Fl",      "ffl",    64260)
325 CHAR("AE",      "AE",      198)
326 CHAR("ae",      "ae",      230)
327 CHAR("OE",      "OE",      338)
328 CHAR("oe",      "oe",      339)
329 CHAR("ss",      "ss",      223)
330 CHAR("IJ",      "IJ",      306)
331 CHAR("ij",      "ij",      307)

333 /* Special letters. */
334 CHAR("-D",      "D",      208)
335 CHAR("sd",      "o",      240)
336 CHAR("TP",      "b",      222)
337 CHAR("Tp",      "b",      254)
338 CHAR(".i",      "i",      305)
339 CHAR(".j",      "j",      567)

341 /* Currency. */
342 CHAR("Do",      "$",      36)
343 CHAR("ct",      "c",      162)
344 CHAR("Eu",      "EUR",    8364)
345 CHAR("eu",      "EUR",    8364)
346 CHAR("Ye",      "Y",      165)
347 CHAR("Po",      "L",      163)
348 CHAR("Cs",      "x",      164)
349 CHAR("Fn",      "f",      402)

351 /* Lines. */
352 CHAR("ba",      "|",      124)
353 CHAR("br",      "|",      9474)
354 CHAR("ul",      "|",      95)
355 CHAR("xl",      "-|",    8254)
356 CHAR("bb",      "|",      166)
357 CHAR("sl",      "/|",     47)
358 CHAR("rs",      "\\|",    92)

360 /* Text markers. */
361 CHAR("ci",      "o",      9675)
362 CHAR("bu",      "o",      8226)
363 CHAR("dd",      "=",      8225)
364 CHAR("dg",      "=",      8224)
365 CHAR("lz",      "<>",    9674)
366 CHAR("sq",      "[ ]",    9633)
367 CHAR("ps",      "9|",    182)
368 CHAR("sc",      "S",      167)
369 CHAR("lh",      "<=",    9756)
370 CHAR("rh",      "=>",    9758)
371 CHAR("at",      "@",      64)
372 CHAR("sh",      "#",      35)
373 CHAR("CR",      "_|",    8629)
374 CHAR("OK",      "\\/",    10003)

376 /* Legal symbols. */
377 CHAR("co",      "(C)",    169)
378 CHAR("rg",      "(R)",    174)
379 CHAR("tm",      "tm",    8482)

381 /* Punctuation. */
382 CHAR(".",       ".",      46)
383 CHAR("r!",      "i",      161)
384 CHAR("r?",      "c",      191)
385 CHAR("em",      "--",    8212)
386 CHAR("en",      "-_",    8211)

```

```
387 CHAR("hy",      "- ",      8208)
388 CHAR("e",       "\\ ",      92)
390 /* Units. */
391 CHAR("de",       "o",      176)
392 CHAR("%0",       "%o",      8240)
393 CHAR("fm",       "\' ",      8242)
394 CHAR("sd",       "\" ",      8243)
395 CHAR("mc",       "mu",      181)
397 CHAR_TBL_END
```

```

*****
1208 Sat Jul 19 19:24:29 2014
new/usr/src/cmd/mandoc/config.h
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****

```

```

1 #ifndef MANDOC_CONFIG_H
2 #define MANDOC_CONFIG_H

4 #if defined(__linux__) || defined(__MINT__)
5 # define _GNU_SOURCE /* strptime(), getsubopt() */
6 #endif

8 #include <stdio.h>

10 #define HAVE_STRPTIME
11 #define HAVE_GETSUBOPT
12 #define HAVE_STRLCAT
13 #define HAVE_STRLCPY

15 #include <sys/types.h>

17 #if !defined(__BEGIN_DECLS)
18 # ifdef __cplusplus
19 # define __BEGIN_DECLS extern "C" {
20 # else
21 # define __BEGIN_DECLS
22 # endif
23 #endif
24 #if !defined(__END_DECLS)
25 # ifdef __cplusplus
26 # define __END_DECLS }
27 # else
28 # define __END_DECLS
29 # endif
30 #endif

32 #if defined(__APPLE__)
33 # define htobe32(x) OSSwapHostToBigInt32(x)
34 # define betoh32(x) OSSwapBigToHostInt32(x)
35 # define htobe64(x) OSSwapHostToBigInt64(x)
36 # define betoh64(x) OSSwapBigToHostInt64(x)
37 #elif defined(__linux__)
38 # define betoh32(x) be32toh(x)
39 # define betoh64(x) be64toh(x)
40 #endif

42 #ifndef HAVE_STRLCAT
43 extern size_t strlcat(char *, const char *, size_t);
44 #endif
45 #ifndef HAVE_STRLCPY
46 extern size_t strlcpy(char *, const char *, size_t);
47 #endif
48 #ifndef HAVE_GETSUBOPT
49 extern int getsubopt(char **, char * const *, char **);
50 extern char *suboptarg;
51 #endif
52 #ifndef HAVE_FGETLN
53 extern char *fgetln(FILE *, size_t *);
54 #endif

56 #endif /* MANDOC_CONFIG_H */

```

```
*****
```

```
21161 Sat Jul 19 19:24:29 2014
```

```
new/usr/src/cmd/mandoc/eqn.c
```

```
5025 import and use mandoc
```

```
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
```

```
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
```

```
Reviewed by: Robert Mustacchi <rm@joyent.com>
```

```
Reviewed by: Albert Lee <trisk@nexenta.com>
```

```
Approved by: TBD
```

```
*****
```

```
1 /* $Id: eqn.c,v 1.38 2011/07/25 15:37:00 kristaps Exp $ */
2 /*
3  * Copyright (c) 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <assert.h>
22 #include <limits.h>
23 #include <stdio.h>
24 #include <stdlib.h>
25 #include <string.h>
26 #include <time.h>
27
28 #include "mandoc.h"
29 #include "libmandoc.h"
30 #include "libroff.h"
31
32 #define EQN_NEST_MAX 128 /* maximum nesting of defines */
33 #define EQN_MSG(t, x) mandoc_msg((t), (x)->parse, (x)->eqn.ln, (x)->eqn.pos,
34
35 enum eqn_rest {
36     EQN_DESCOPE,
37     EQN_ERR,
38     EQN_OK,
39     EQN_EOF
40 };
41
42 enum eqn_synt {
43     EQNSYM_alpha,
44     EQNSYM_beta,
45     EQNSYM_chi,
46     EQNSYM_delta,
47     EQNSYM_epsilon,
48     EQNSYM_eta,
49     EQNSYM_gamma,
50     EQNSYM_iota,
51     EQNSYM_kappa,
52     EQNSYM_lambda,
53     EQNSYM_mu,
54     EQNSYM_nu,
55     EQNSYM_omega,
56     EQNSYM_omicron,
```

```
57     EQNSYM_phi,
58     EQNSYM_pi,
59     EQNSYM_ps,
60     EQNSYM_rho,
61     EQNSYM_sigma,
62     EQNSYM_tau,
63     EQNSYM_theta,
64     EQNSYM_upsilon,
65     EQNSYM_xi,
66     EQNSYM_zeta,
67     EQNSYM_DELTA,
68     EQNSYM_GAMMA,
69     EQNSYM_LAMBDA,
70     EQNSYM_OMEGA,
71     EQNSYM_PHI,
72     EQNSYM_PI,
73     EQNSYM_PSI,
74     EQNSYM_SIGMA,
75     EQNSYM_THETA,
76     EQNSYM_UPSILON,
77     EQNSYM_XI,
78     EQNSYM_inter,
79     EQNSYM_union,
80     EQNSYM_prod,
81     EQNSYM_int,
82     EQNSYM_sum,
83     EQNSYM_grad,
84     EQNSYM_del,
85     EQNSYM_times,
86     EQNSYM_cdot,
87     EQNSYM_nothing,
88     EQNSYM_approx,
89     EQNSYM_prime,
90     EQNSYM_half,
91     EQNSYM_partial,
92     EQNSYM_inf,
93     EQNSYM_muchgreat,
94     EQNSYM_muchless,
95     EQNSYM_larrow,
96     EQNSYM_rarrow,
97     EQNSYM_pm,
98     EQNSYM_nequal,
99     EQNSYM_equiv,
100    EQNSYM_lessequal,
101    EQNSYM_moreequal,
102    EQNSYM_MAX
103 };
104
105 enum eqnpartt {
106     EQN_DEFINE = 0,
107     EQN_UNDEF,
108     EQN_TDEFINE,
109     EQN_SET,
110     EQN_UNDEF,
111     EQN_GFONT,
112     EQN_GSIZE,
113     EQN_BACK,
114     EQN_FWD,
115     EQN_UP,
116     EQN_DOWN,
117     EQN_MAX
118 };
119
120 struct eqnstr {
121     const char *name;
122     size_t sz;
```

```

123 };

125 #define STRNEQ(p1, sz1, p2, sz2) \
126 ((sz1) == (sz2) && 0 == strncmp((p1), (p2), (sz1)))
127 #define EQNSTREQ(x, p, sz) \
128 STRNEQ((x)->name, (x)->sz, (p), (sz))

130 struct eqnpart {
131     struct eqnstr    str;
132     int              (*fp)(struct eqn_node *);
133 };

135 struct eqnsym {
136     struct eqnstr    str;
137     const char       *sym;
138 };

141 static enum eqn_rest eqn_box(struct eqn_node *, struct eqn_box *);
142 static struct eqn_box *eqn_box_alloc(struct eqn_node *,
143     struct eqn_box *);
144 static void eqn_box_free(struct eqn_box *);
145 static struct eqn_def *eqn_def_find(struct eqn_node *,
146     const char *, size_t);
147 static int eqn_do_gfont(struct eqn_node *);
148 static int eqn_do_gsize(struct eqn_node *);
149 static int eqn_do_define(struct eqn_node *);
150 static int eqn_do_ign1(struct eqn_node *);
151 static int eqn_do_ign2(struct eqn_node *);
152 static int eqn_do_tdefine(struct eqn_node *);
153 static int eqn_do_undef(struct eqn_node *);
154 static enum eqn_rest eqn_eqn(struct eqn_node *, struct eqn_box *);
155 static enum eqn_rest eqn_list(struct eqn_node *, struct eqn_box *);
156 static enum eqn_rest eqn_matrix(struct eqn_node *, struct eqn_box *);
157 static const char *eqn_nexttok(struct eqn_node *, size_t *);
158 static const char *eqn_nextrawtok(struct eqn_node *, size_t *);
159 static const char *eqn_next(struct eqn_node *,
160     char, size_t *, int);
161 static void eqn_rewind(struct eqn_node *);

163 static const struct eqnpart eqnparts[EQN_MAX] = {
164     { "define", 6 }, eqn_do_define }, /* EQN_DEFINE */
165     { "ndefine", 7 }, eqn_do_define }, /* EQN_NDEFINE */
166     { "tdefine", 7 }, eqn_do_tdefine }, /* EQN_TDEFINE */
167     { "set", 3 }, eqn_do_ign2 }, /* EQN_SET */
168     { "undef", 5 }, eqn_do_undef }, /* EQN_UNDEF */
169     { "gfont", 5 }, eqn_do_gfont }, /* EQN_GFONT */
170     { "gsize", 5 }, eqn_do_gsize }, /* EQN_GSIZE */
171     { "back", 4 }, eqn_do_ign1 }, /* EQN_BACK */
172     { "fwd", 3 }, eqn_do_ign1 }, /* EQN_FWD */
173     { "up", 2 }, eqn_do_ign1 }, /* EQN_UP */
174     { "down", 4 }, eqn_do_ign1 }, /* EQN_DOWN */
175 };

177 static const struct eqnstr eqnmarks[EQNMARK_MAX] = {
178     { "", 0 }, /* EQNMARK_NONE */
179     { "dot", 3 }, /* EQNMARK_DOT */
180     { "dotdot", 6 }, /* EQNMARK_DOTDOT */
181     { "hat", 3 }, /* EQNMARK_HAT */
182     { "tilde", 5 }, /* EQNMARK_TILDE */
183     { "vec", 3 }, /* EQNMARK_VEC */
184     { "dyad", 4 }, /* EQNMARK_DYAD */
185     { "bar", 3 }, /* EQNMARK_BAR */
186     { "under", 5 }, /* EQNMARK_UNDER */
187 };

```

```

189 static const struct eqnstr eqnfonts[EQNFONT_MAX] = {
190     { "", 0 }, /* EQNFONT_NONE */
191     { "roman", 5 }, /* EQNFONT_ROMAN */
192     { "bold", 4 }, /* EQNFONT_BOLD */
193     { "fat", 3 }, /* EQNFONT_FAT */
194     { "italic", 6 }, /* EQNFONT_ITALIC */
195 };

197 static const struct eqnstr eqnpos[EQNPOS_MAX] = {
198     { "", 0 }, /* EQNPOS_NONE */
199     { "over", 4 }, /* EQNPOS_OVER */
200     { "sup", 3 }, /* EQNPOS_SUP */
201     { "sub", 3 }, /* EQNPOS_SUB */
202     { "to", 2 }, /* EQNPOS_TO */
203     { "from", 4 }, /* EQNPOS_FROM */
204 };

206 static const struct eqnstr eqnpiles[EQNPILE_MAX] = {
207     { "", 0 }, /* EQNPILE_NONE */
208     { "pile", 4 }, /* EQNPILE_PILE */
209     { "cpile", 5 }, /* EQNPILE_CPILE */
210     { "rpile", 5 }, /* EQNPILE_RPILE */
211     { "lpile", 5 }, /* EQNPILE_LPILE */
212     { "col", 3 }, /* EQNPILE_COL */
213     { "ccol", 4 }, /* EQNPILE_CCOL */
214     { "rcol", 4 }, /* EQNPILE_RCOL */
215     { "lcol", 4 }, /* EQNPILE_LCOL */
216 };

218 static const struct eqnsym eqnsyms[EQNSYM_MAX] = {
219     { "alpha", 5 }, /* EQNSYM_alpha */
220     { "beta", 4 }, /* EQNSYM_beta */
221     { "chi", 3 }, /* EQNSYM_chi */
222     { "delta", 5 }, /* EQNSYM_delta */
223     { "epsilon", 7 }, /* EQNSYM_epsilon */
224     { "eta", 3 }, /* EQNSYM_eta */
225     { "gamma", 5 }, /* EQNSYM_gamma */
226     { "iota", 4 }, /* EQNSYM_iota */
227     { "kappa", 5 }, /* EQNSYM_kappa */
228     { "lambda", 6 }, /* EQNSYM_lambda */
229     { "mu", 2 }, /* EQNSYM_mu */
230     { "nu", 2 }, /* EQNSYM_nu */
231     { "omega", 5 }, /* EQNSYM_omega */
232     { "omicron", 7 }, /* EQNSYM_omicron */
233     { "phi", 3 }, /* EQNSYM_phi */
234     { "pi", 2 }, /* EQNSYM_pi */
235     { "psi", 2 }, /* EQNSYM_psi */
236     { "rho", 3 }, /* EQNSYM_rho */
237     { "sigma", 5 }, /* EQNSYM_sigma */
238     { "tau", 3 }, /* EQNSYM_tau */
239     { "theta", 5 }, /* EQNSYM_theta */
240     { "upsilon", 7 }, /* EQNSYM_upsilon */
241     { "xi", 2 }, /* EQNSYM_xi */
242     { "zeta", 4 }, /* EQNSYM_zeta */
243     { "DELTA", 5 }, /* EQNSYM_DELTA */
244     { "GAMMA", 5 }, /* EQNSYM_GAMMA */
245     { "LAMBDA", 6 }, /* EQNSYM_LAMBDA */
246     { "OMEGA", 5 }, /* EQNSYM_OMEGA */
247     { "PHI", 3 }, /* EQNSYM_PHI */
248     { "PI", 2 }, /* EQNSYM_PI */
249     { "PSI", 3 }, /* EQNSYM_PSI */
250     { "SIGMA", 5 }, /* EQNSYM_SIGMA */
251     { "THETA", 5 }, /* EQNSYM_THETA */
252     { "UPSILON", 7 }, /* EQNSYM_UPSILON */
253     { "XI", 2 }, /* EQNSYM_XI */
254     { "inter", 5 }, /* EQNSYM_inter */

```

```

255 { "union", 5 }, "cu" }, /* EQNSYM_union */
256 { "prod", 4 }, "product" }, /* EQNSYM_prod */
257 { "int", 3 }, "integral" }, /* EQNSYM_int */
258 { "sum", 3 }, "sum" }, /* EQNSYM_sum */
259 { "grad", 4 }, "gr" }, /* EQNSYM_grad */
260 { "del", 3 }, "gr" }, /* EQNSYM_del */
261 { "times", 5 }, "mu" }, /* EQNSYM_times */
262 { "cdot", 4 }, "pc" }, /* EQNSYM_cdot */
263 { "nothing", 7 }, "&" }, /* EQNSYM_nothing */
264 { "approx", 6 }, "~=" }, /* EQNSYM_approx */
265 { "prime", 5 }, "aq" }, /* EQNSYM_prime */
266 { "half", 4 }, "l2" }, /* EQNSYM_half */
267 { "partial", 7 }, "pd" }, /* EQNSYM_partial */
268 { "inf", 3 }, "if" }, /* EQNSYM_inf */
269 { ">>", 2 }, ">>" }, /* EQNSYM_muchgreat */
270 { "<<", 2 }, "<<" }, /* EQNSYM_muchless */
271 { "<=", 2 }, "<=" }, /* EQNSYM_larrow */
272 { ">=", 2 }, ">=" }, /* EQNSYM_rarrow */
273 { "+-", 2 }, "+-" }, /* EQNSYM_pm */
274 { "!=", 2 }, "!=" }, /* EQNSYM_nequal */
275 { "==", 2 }, "==" }, /* EQNSYM_equiv */
276 { "<=", 2 }, "<=" }, /* EQNSYM_lessequal */
277 { ">=", 2 }, ">=" }, /* EQNSYM_moreequal */
278 };

280 /* ARGSUSED */
281 enum rofferr
282 eqn_read(struct eqn_node **ep, int ln,
283          const char *p, int pos, int *offs)
284 {
285     size_t      sz;
286     struct eqn_node *ep;
287     enum rofferr  er;

289     ep = *ep;

291     /*
292      * If we're the terminating mark, unset our equation status and
293      * validate the full equation.
294      */

296     if (0 == strcmp(p, ".EN", 3)) {
297         er = eqn_end(ep);
298         p += 3;
299         while (' ' == *p || '\t' == *p)
300             p++;
301         if ('\0' == *p)
302             return(er);
303         mandoc_msg(MANDOCERR_ARGSLIST, ep->parse, ln, pos, NULL);
304         return(er);
305     }

307     /*
308      * Build up the full string, replacing all newlines with regular
309      * whitespace.
310      */

312     sz = strlen(p + pos) + 1;
313     ep->data = mandoc_realloc(ep->data, ep->sz + sz + 1);

315     /* First invocation: nil terminate the string. */

317     if (0 == ep->sz)
318         *ep->data = '\0';

320     ep->sz += sz;

```

```

321     strcat(ep->data, p + pos, ep->sz + 1);
322     strcat(ep->data, " ", ep->sz + 1);
323     return(ROFF_IGN);
324 }

326 struct eqn_node *
327 eqn_alloc(const char *name, int pos, int line, struct mparse *parse)
328 {
329     struct eqn_node *p;
330     size_t      sz;
331     const char  *end;

333     p = mandoc_calloc(1, sizeof(struct eqn_node));

335     if (name && '\0' != *name) {
336         sz = strlen(name);
337         assert(sz);
338         do {
339             sz--;
340             end = name + (int)sz;
341         } while (' ' == *end || '\t' == *end);
342         p->eqn.name = mandoc_strdup(name, sz + 1);
343     }

345     p->parse = parse;
346     p->eqn.ln = line;
347     p->eqn.pos = pos;
348     p->gsize = EQN_DEFSIZE;

350     return(p);
351 }

353 enum rofferr
354 eqn_end(struct eqn_node **ep)
355 {
356     struct eqn_node *ep;
357     struct eqn_box  *root;
358     enum eqn_rest    c;

360     ep = *ep;
361     *ep = NULL;

363     ep->eqn.root = mandoc_calloc(1, sizeof(struct eqn_box));

365     root = ep->eqn.root;
366     root->type = EQN_ROOT;

368     if (0 == ep->sz)
369         return(ROFF_IGN);

371     if (EQN_DESCOPE == (c = eqn_eqn(ep, root))) {
372         EQN_MSG(MANDOCERR_EQNNSCOPE, ep);
373         c = EQN_ERR;
374     }

376     return(EQN_EOF == c ? ROFF_EQN : ROFF_IGN);
377 }

379 static enum eqn_rest
380 eqn_eqn(struct eqn_node *ep, struct eqn_box *last)
381 {
382     struct eqn_box  *bp;
383     enum eqn_rest    c;

385     bp = eqn_box_alloc(ep, last);
386     bp->type = EQN_SUBEXPR;

```

```

388     while (EQN_OK == (c = eqn_box(ep, bp)))
389         /* Spin! */ ;
391     return(c);
392 }
394 static enum eqn_rest
395 eqn_matrix(struct eqn_node *ep, struct eqn_box *last)
396 {
397     struct eqn_box *bp;
398     const char *start;
399     size_t sz;
400     enum eqn_rest c;
402     bp = eqn_box_alloc(ep, last);
403     bp->type = EQN_MATRIX;
405     if (NULL == (start = eqn_nexttok(ep, &sz))) {
406         EQN_MSG(MANDOCERR_EQNEOF, ep);
407         return(EQN_ERR);
408     }
409     if ( ! STRNEQ(start, sz, "{", 1)) {
410         EQN_MSG(MANDOCERR_EQNSYNT, ep);
411         return(EQN_ERR);
412     }
414     while (EQN_OK == (c = eqn_box(ep, bp)))
415         switch (bp->last->pile) {
416             case (EQNPILE_LCOL):
417                 /* FALLTHROUGH */
418             case (EQNPILE_CCOL):
419                 /* FALLTHROUGH */
420             case (EQNPILE_RCOL):
421                 continue;
422             default:
423                 EQN_MSG(MANDOCERR_EQNSYNT, ep);
424                 return(EQN_ERR);
425         };
427     if (EQN_DESCOPE != c) {
428         if (EQN_EOF == c)
429             EQN_MSG(MANDOCERR_EQNEOF, ep);
430         return(EQN_ERR);
431     }
433     eqn_rewind(ep);
434     start = eqn_nexttok(ep, &sz);
435     assert(start);
436     if (STRNEQ(start, sz, "}", 1))
437         return(EQN_OK);
439     EQN_MSG(MANDOCERR_EQNBADSCOPE, ep);
440     return(EQN_ERR);
441 }
443 static enum eqn_rest
444 eqn_list(struct eqn_node *ep, struct eqn_box *last)
445 {
446     struct eqn_box *bp;
447     const char *start;
448     size_t sz;
449     enum eqn_rest c;
451     bp = eqn_box_alloc(ep, last);
452     bp->type = EQN_LIST;

```

```

454     if (NULL == (start = eqn_nexttok(ep, &sz))) {
455         EQN_MSG(MANDOCERR_EQNEOF, ep);
456         return(EQN_ERR);
457     }
458     if ( ! STRNEQ(start, sz, "{", 1)) {
459         EQN_MSG(MANDOCERR_EQNSYNT, ep);
460         return(EQN_ERR);
461     }
463     while (EQN_DESCOPE == (c = eqn_eqn(ep, bp))) {
464         eqn_rewind(ep);
465         start = eqn_nexttok(ep, &sz);
466         assert(start);
467         if ( ! STRNEQ(start, sz, "above", 5))
468             break;
469     }
471     if (EQN_DESCOPE != c) {
472         if (EQN_ERR != c)
473             EQN_MSG(MANDOCERR_EQNSCOPE, ep);
474         return(EQN_ERR);
475     }
477     eqn_rewind(ep);
478     start = eqn_nexttok(ep, &sz);
479     assert(start);
480     if (STRNEQ(start, sz, "}", 1))
481         return(EQN_OK);
483     EQN_MSG(MANDOCERR_EQNBADSCOPE, ep);
484     return(EQN_ERR);
485 }
487 static enum eqn_rest
488 eqn_box(struct eqn_node *ep, struct eqn_box *last)
489 {
490     size_t sz;
491     const char *start;
492     char *left;
493     char sym[64];
494     enum eqn_rest c;
495     int i, size;
496     struct eqn_box *bp;
498     if (NULL == (start = eqn_nexttok(ep, &sz)))
499         return(EQN_EOF);
501     if (STRNEQ(start, sz, "}", 1))
502         return(EQN_DESCOPE);
503     else if (STRNEQ(start, sz, "right", 5))
504         return(EQN_DESCOPE);
505     else if (STRNEQ(start, sz, "above", 5))
506         return(EQN_DESCOPE);
507     else if (STRNEQ(start, sz, "mark", 4))
508         return(EQN_OK);
509     else if (STRNEQ(start, sz, "lineup", 6))
510         return(EQN_OK);
512     for (i = 0; i < (int)EQN_MAX; i++) {
513         if ( ! EQNSTREQ(&eqnparts[i].str, start, sz))
514             continue;
515         return((*eqnparts[i].fp)(ep) ?
516             EQN_OK : EQN_ERR);
517     }

```



```

519     if (STRNEQ(start, sz, "{", 1)) {
520         if (EQN_DESCOPE != (c = eqn_eqn(ep, last))) {
521             if (EQN_ERR != c)
522                 EQN_MSG(MANDOCERR_EQNSCOPE, ep);
523             return(EQN_ERR);
524         }
525         eqn_rewind(ep);
526         start = eqn_nexttok(ep, &sz);
527         assert(start);
528         if (STRNEQ(start, sz, "}", 1))
529             return(EQN_OK);
530         EQN_MSG(MANDOCERR_EQNBADSCOPE, ep);
531         return(EQN_ERR);
532     }

534     for (i = 0; i < (int)EQNPILE_MAX; i++) {
535         if (!EQNSTREQ(&eqnpiles[i], start, sz))
536             continue;
537         if (EQN_OK == (c = eqn_list(ep, last)))
538             last->last->pile = (enum eqn_pilet)i;
539         return(c);
540     }

542     if (STRNEQ(start, sz, "matrix", 6))
543         return(eqn_matrix(ep, last));

545     if (STRNEQ(start, sz, "left", 4)) {
546         if (NULL == (start = eqn_nexttok(ep, &sz))) {
547             EQN_MSG(MANDOCERR_EQNEOF, ep);
548             return(EQN_ERR);
549         }
550         left = mandoc_strndup(start, sz);
551         c = eqn_eqn(ep, last);
552         if (last->last)
553             last->last->left = left;
554         else
555             free(left);
556         if (EQN_DESCOPE != c)
557             return(c);
558         assert(last->last);
559         eqn_rewind(ep);
560         start = eqn_nexttok(ep, &sz);
561         assert(start);
562         if (!STRNEQ(start, sz, "right", 5))
563             return(EQN_DESCOPE);
564         if (NULL == (start = eqn_nexttok(ep, &sz))) {
565             EQN_MSG(MANDOCERR_EQNEOF, ep);
566             return(EQN_ERR);
567         }
568         last->last->right = mandoc_strndup(start, sz);
569         return(EQN_OK);
570     }

572     for (i = 0; i < (int)EQNPOS_MAX; i++) {
573         if (!EQNSTREQ(&eqnpos[i], start, sz))
574             continue;
575         if (NULL == last->last) {
576             EQN_MSG(MANDOCERR_EQNSYNT, ep);
577             return(EQN_ERR);
578         }
579         last->last->pos = (enum eqn_post)i;
580         if (EQN_EOF == (c = eqn_box(ep, last))) {
581             EQN_MSG(MANDOCERR_EQNEOF, ep);
582             return(EQN_ERR);
583         }
584         return(c);

```

```

585     }

587     for (i = 0; i < (int)EQNMARK_MAX; i++) {
588         if (!EQNSTREQ(&eqnmarks[i], start, sz))
589             continue;
590         if (NULL == last->last) {
591             EQN_MSG(MANDOCERR_EQNSYNT, ep);
592             return(EQN_ERR);
593         }
594         last->last->mark = (enum eqn_markt)i;
595         if (EQN_EOF == (c = eqn_box(ep, last))) {
596             EQN_MSG(MANDOCERR_EQNEOF, ep);
597             return(EQN_ERR);
598         }
599         return(c);
600     }

602     for (i = 0; i < (int)EQNFONT_MAX; i++) {
603         if (!EQNSTREQ(&eqnfonts[i], start, sz))
604             continue;
605         if (EQN_EOF == (c = eqn_box(ep, last))) {
606             EQN_MSG(MANDOCERR_EQNEOF, ep);
607             return(EQN_ERR);
608         } else if (EQN_OK == c)
609             last->last->font = (enum eqn_fontt)i;
610         return(c);
611     }

613     if (STRNEQ(start, sz, "size", 4)) {
614         if (NULL == (start = eqn_nexttok(ep, &sz))) {
615             EQN_MSG(MANDOCERR_EQNEOF, ep);
616             return(EQN_ERR);
617         }
618         size = mandoc_strntoi(start, sz, 10);
619         if (EQN_EOF == (c = eqn_box(ep, last))) {
620             EQN_MSG(MANDOCERR_EQNEOF, ep);
621             return(EQN_ERR);
622         } else if (EQN_OK != c)
623             return(c);
624         last->last->size = size;
625     }

627     bp = eqn_box_alloc(ep, last);
628     bp->type = EQN_TEXT;
629     for (i = 0; i < (int)EQNSYM_MAX; i++)
630         if (EQNSTREQ(&eqnsyms[i].str, start, sz)) {
631             sym[63] = '\0';
632             snprintf(sym, 62, "\\[%s]", eqnsyms[i].sym);
633             bp->text = mandoc_strdup(sym);
634             return(EQN_OK);
635         }

637     bp->text = mandoc_strndup(start, sz);
638     return(EQN_OK);
639 }

641 void
642 eqn_free(struct eqn_node *p)
643 {
644     int i;

646     eqn_box_free(p->eqn.root);

648     for (i = 0; i < (int)p->defsz; i++) {
649         free(p->defs[i].key);
650         free(p->defs[i].val);

```

```

651     }
653     free(p->eqn.name);
654     free(p->data);
655     free(p->defs);
656     free(p);
657 }

659 static struct eqn_box *
660 eqn_box_alloc(struct eqn_node *ep, struct eqn_box *parent)
661 {
662     struct eqn_box *bp;

664     bp = mandoc_calloc(1, sizeof(struct eqn_box));
665     bp->parent = parent;
666     bp->size = ep->gsize;

668     if (NULL == parent->first)
669         parent->first = bp;
670     else
671         parent->last->next = bp;

673     parent->last = bp;
674     return(bp);
675 }

677 static void
678 eqn_box_free(struct eqn_box *bp)
679 {
681     if (bp->first)
682         eqn_box_free(bp->first);
683     if (bp->next)
684         eqn_box_free(bp->next);

686     free(bp->text);
687     free(bp->left);
688     free(bp->right);
689     free(bp);
690 }

692 static const char *
693 eqn_nextrawtok(struct eqn_node *ep, size_t *sz)
694 {
696     return(eqn_next(ep, "", sz, 0));
697 }

699 static const char *
700 eqn_nexttok(struct eqn_node *ep, size_t *sz)
701 {
703     return(eqn_next(ep, "", sz, 1));
704 }

706 static void
707 eqn_rewind(struct eqn_node *ep)
708 {
710     ep->cur = ep->rew;
711 }

713 static const char *
714 eqn_next(struct eqn_node *ep, char quote, size_t *sz, int repl)
715 {
716     char          *start, *next;

```

```

717     int          q, diff, lim;
718     size_t      ssz, dummy;
719     struct eqn_def *def;

721     if (NULL == sz)
722         sz = &dummy;

724     lim = 0;
725     ep->rew = ep->cur;
726 again:
727     /* Prevent self-definitions. */

729     if (lim >= EQN_NEST_MAX) {
730         EQN_MSG(MANDOCERR_ROFFLOOP, ep);
731         return(NULL);
732     }

734     ep->cur = ep->rew;
735     start = &ep->data[(int)ep->cur];
736     q = 0;

738     if ('\0' == *start)
739         return(NULL);

741     if (quote == *start) {
742         ep->cur++;
743         q = 1;
744     }

746     start = &ep->data[(int)ep->cur];

748     if ( ! q) {
749         if ('{' == *start || '\'' == *start)
750             ssz = 1;
751         else
752             ssz = strcspn(start + 1, " ^~\n{ }\t" ) + 1;
753         next = start + (int)ssz;
754         if ('\0' == *next)
755             next = NULL;
756     } else
757         next = strchr(start, quote);

759     if (NULL != next) {
760         *sz = (size_t)(next - start);
761         ep->cur += *sz;
762         if (q)
763             ep->cur++;
764         while (' ' == ep->data[(int)ep->cur] ||
765                '\t' == ep->data[(int)ep->cur] ||
766                '^' == ep->data[(int)ep->cur] ||
767                '~' == ep->data[(int)ep->cur])
768             ep->cur++;
769     } else {
770         if (q)
771             EQN_MSG(MANDOCERR_BADQUOTE, ep);
772         next = strchr(start, '\0');
773         *sz = (size_t)(next - start);
774         ep->cur += *sz;
775     }

777     /* Quotes aren't expanded for values. */

779     if (q || ! repl)
780         return(start);

782     if (NULL != (def = eqn_def_find(ep, start, *sz))) {

```

```

783     diff = def->valsz - *sz;
785     if (def->valsz > *sz) {
786         ep->sz += diff;
787         ep->data = mandoc_realloc(ep->data, ep->sz + 1);
788         ep->data[ep->sz] = '\0';
789         start = &ep->data[(int)ep->rew];
790     }
792     diff = def->valsz - *sz;
793     memmove(start + *sz + diff, start + *sz,
794             (strlen(start) - *sz) + 1);
795     memcpy(start, def->val, def->valsz);
796     goto again;
797 }
799     return(start);
800 }
802 static int
803 eqn_do_ign1(struct eqn_node *ep)
804 {
806     if (NULL == eqn_nextrawtok(ep, NULL))
807         EQN_MSG(MANDOCERR_EQNEOF, ep);
808     else
809         return(1);
811     return(0);
812 }
814 static int
815 eqn_do_ign2(struct eqn_node *ep)
816 {
818     if (NULL == eqn_nextrawtok(ep, NULL))
819         EQN_MSG(MANDOCERR_EQNEOF, ep);
820     else if (NULL == eqn_nextrawtok(ep, NULL))
821         EQN_MSG(MANDOCERR_EQNEOF, ep);
822     else
823         return(1);
825     return(0);
826 }
828 static int
829 eqn_do_tdefine(struct eqn_node *ep)
830 {
832     if (NULL == eqn_nextrawtok(ep, NULL))
833         EQN_MSG(MANDOCERR_EQNEOF, ep);
834     else if (NULL == eqn_next(ep, ep->data[(int)ep->cur], NULL, 0))
835         EQN_MSG(MANDOCERR_EQNEOF, ep);
836     else
837         return(1);
839     return(0);
840 }
842 static int
843 eqn_do_define(struct eqn_node *ep)
844 {
845     const char    *start;
846     size_t        sz;
847     struct eqn_def *def;
848     int           i;

```

```

850     if (NULL == (start = eqn_nextrawtok(ep, &sz))) {
851         EQN_MSG(MANDOCERR_EQNEOF, ep);
852         return(0);
853     }
855     /*
856     * Search for a key that already exists.
857     * Create a new key if none is found.
858     */
860     if (NULL == (def = eqn_def_find(ep, start, sz))) {
861         /* Find holes in string array. */
862         for (i = 0; i < (int)ep->defsz; i++)
863             if (0 == ep->defs[i].keylen)
864                 break;
866         if (i == (int)ep->defsz) {
867             ep->defsz++;
868             ep->defs = mandoc_realloc
869                 (ep->defs, ep->defsz *
870                 sizeof(struct eqn_def));
871             ep->defs[i].key = ep->defs[i].val = NULL;
872         }
874         ep->defs[i].keylen = sz;
875         ep->defs[i].key = mandoc_realloc
876             (ep->defs[i].key, sz + 1);
878         memcpy(ep->defs[i].key, start, sz);
879         ep->defs[i].key[(int)sz] = '\0';
880         def = &ep->defs[i];
881     }
883     start = eqn_next(ep, ep->data[(int)ep->cur], &sz, 0);
885     if (NULL == start) {
886         EQN_MSG(MANDOCERR_EQNEOF, ep);
887         return(0);
888     }
890     def->valsz = sz;
891     def->val = mandoc_realloc(def->val, sz + 1);
892     memcpy(def->val, start, sz);
893     def->val[(int)sz] = '\0';
894     return(1);
895 }
897 static int
898 eqn_do_gfont(struct eqn_node *ep)
899 {
901     if (NULL == eqn_nextrawtok(ep, NULL)) {
902         EQN_MSG(MANDOCERR_EQNEOF, ep);
903         return(0);
904     }
905     return(1);
906 }
908 static int
909 eqn_do_gsize(struct eqn_node *ep)
910 {
911     const char    *start;
912     size_t        sz;
914     if (NULL == (start = eqn_nextrawtok(ep, &sz))) {

```

```
915         EQN_MSG(MANDOCERR_EQNEOF, ep);
916         return(0);
917     }
918     ep->gsize = mandoc_strntoi(start, sz, 10);
919     return(1);
920 }

922 static int
923 eqn_do_undef(struct eqn_node *ep)
924 {
925     const char    *start;
926     struct eqn_def *def;
927     size_t        sz;

929     if (NULL == (start = eqn_nextrawtok(ep, &sz))) {
930         EQN_MSG(MANDOCERR_EQNEOF, ep);
931         return(0);
932     } else if (NULL != (def = eqn_def_find(ep, start, sz)))
933         def->keysz = 0;

935     return(1);
936 }

938 static struct eqn_def *
939 eqn_def_find(struct eqn_node *ep, const char *key, size_t sz)
940 {
941     int            i;

943     for (i = 0; i < (int)ep->defsz; i++)
944         if (ep->defs[i].keysz && STRNEQ(ep->defs[i].key,
945             ep->defs[i].keysz, key, sz))
946             return(&ep->defs[i]);

948     return(NULL);
949 }
```

new/usr/src/cmd/mandoc/eqn_html.c

1

1996 Sat Jul 19 19:24:29 2014
new/usr/src/cmd/mandoc/eqn_html.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD

```
1 /* $Id: eqn_html.c,v 1.2 2011/07/24 10:09:03 kristaps Exp $ */
2 /*
3  * Copyright (c) 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <assert.h>
22 #include <stdio.h>
23 #include <stdlib.h>
24 #include <string.h>
25
26 #include "mandoc.h"
27 #include "out.h"
28 #include "html.h"
29
30 static const enum htmltag fontmap[EQNFONT__MAX] = {
31     TAG_SPAN, /* EQNFONT_NONE */
32     TAG_SPAN, /* EQNFONT_ROMAN */
33     TAG_B, /* EQNFONT_BOLD */
34     TAG_B, /* EQNFONT_FAT */
35     TAG_I /* EQNFONT_ITALIC */
36 };
37
38
39 static void eqn_box(struct html *, const struct eqn_box *);
40
41 void
42 print_eqn(struct html *p, const struct eqn *ep)
43 {
44     struct htmlpair tag;
45     struct tag *t;
46
47     PAIR_CLASS_INIT(&tag, "eqn");
48     t = print_otag(p, TAG_SPAN, 1, &tag);
49
50     p->flags |= HTML_NONOSPACE;
51     eqn_box(p, ep->root);
52     p->flags &= ~HTML_NONOSPACE;
53
54     print_tagq(p, t);
55 }
```

new/usr/src/cmd/mandoc/eqn_html.c

2

```
57 static void
58 eqn_box(struct html *p, const struct eqn_box *bp)
59 {
60     struct tag *t;
61
62     t = EQNFONT_NONE == bp->font ? NULL :
63         print_otag(p, fontmap[(int)bp->font], 0, NULL);
64
65     if (bp->left)
66         print_text(p, bp->left);
67
68     if (bp->text)
69         print_text(p, bp->text);
70
71     if (bp->first)
72         eqn_box(p, bp->first);
73
74     if (NULL != t)
75         print_tagq(p, t);
76     if (bp->right)
77         print_text(p, bp->right);
78
79     if (bp->next)
80         eqn_box(p, bp->next);
81 }
```

```
*****
```

```
2006 Sat Jul 19 19:24:29 2014
new/usr/src/cmd/mandoc/eqn_term.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
```

```
1 /* $Id: eqn_term.c,v 1.4 2011/07/24 10:09:03 kristaps Exp $ */
2 /*
3  * Copyright (c) 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <assert.h>
22 #include <stdio.h>
23 #include <stdlib.h>
24 #include <string.h>
25
26 #include "mandoc.h"
27 #include "out.h"
28 #include "term.h"
29
30 static const enum termfont fontmap[EQNFONT__MAX] = {
31     TERMFONT_NONE, /* EQNFONT_NONE */
32     TERMFONT_NONE, /* EQNFONT_ROMAN */
33     TERMFONT_BOLD, /* EQNFONT_BOLD */
34     TERMFONT_BOLD, /* EQNFONT_FAT */
35     TERMFONT_UNDER /* EQNFONT_ITALIC */
36 };
37
38 static void eqn_box(struct term *t, const struct eqn_box *b);
39
40 void
41 term_eqn(struct term *t, const struct eqn *e)
42 {
43     struct eqn_box b;
44     b.p = t;
45     b.ep = e;
46     b.flags |= TERMP_NONOSPACE;
47     eqn_box(t, b);
48     b.flags &= ~TERMP_NONOSPACE;
49 }
50
51 static void
52 eqn_box(struct term *t, const struct eqn_box *b)
53 {
54     if (EQNFONT_NONE != b->font)
55         term_fontpush(t, fontmap[(int)b->font]);
56     if (b->left)
```

```
57         term_word(p, b->left);
58         if (EQN_SUBEXPR == b->type)
59             term_word(p, "(");
60
61         if (b->text)
62             term_word(p, b->text);
63
64         if (b->first)
65             eqn_box(p, b->first);
66
67         if (EQN_SUBEXPR == b->type)
68             term_word(p, ")");
69         if (b->right)
70             term_word(p, b->right);
71         if (EQNFONT_NONE != b->font)
72             term_fontpop(p);
73
74         if (b->next)
75             eqn_box(p, b->next);
76     }
```

```

*****
14142 Sat Jul 19 19:24:29 2014
new/usr/src/cmd/mandoc/html.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: html.c,v 1.150 2011/10/05 21:35:17 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <sys/types.h>
23
24 #include <assert.h>
25 #include <ctype.h>
26 #include <stdarg.h>
27 #include <stdio.h>
28 #include <stdint.h>
29 #include <stdlib.h>
30 #include <string.h>
31 #include <unistd.h>
32
33 #include "mandoc.h"
34 #include "libmandoc.h"
35 #include "out.h"
36 #include "html.h"
37 #include "main.h"
38
39 struct htldata {
40     const char    *name;
41     int           flags;
42 #define HTML_CLRLINE    (1 << 0)
43 #define HTML_NOSTACK    (1 << 1)
44 #define HTML_AUTOCLOSE  (1 << 2) /* Tag has auto-closure. */
45 };
46
47 static const struct htldata htmltags[TAG_MAX] = {
48     {"html",      HTML_CLRLINE}, /* TAG_HTML */
49     {"head",     HTML_CLRLINE}, /* TAG_HEAD */
50     {"body",     HTML_CLRLINE}, /* TAG_BODY */
51     {"meta",     HTML_CLRLINE | HTML_NOSTACK | HTML_AUTOCLOSE}, /* TAG_ME
52     {"title",    HTML_CLRLINE}, /* TAG_TITLE */
53     {"div",      HTML_CLRLINE}, /* TAG_DIV */
54     {"h1",       0}, /* TAG_H1 */
55     {"h2",       0}, /* TAG_H2 */
56     {"span",     0}, /* TAG_SPAN */

```

```

57     {"link",     HTML_CLRLINE | HTML_NOSTACK | HTML_AUTOCLOSE}, /* TAG_LI
58     {"br",      HTML_CLRLINE | HTML_NOSTACK | HTML_AUTOCLOSE}, /* TAG_BR
59     {"a",       0}, /* TAG_A */
60     {"table",   HTML_CLRLINE}, /* TAG_TABLE */
61     {"tbody",   HTML_CLRLINE}, /* TAG_TBODY */
62     {"col",     HTML_CLRLINE | HTML_NOSTACK | HTML_AUTOCLOSE}, /* TAG_CO
63     {"tr",      HTML_CLRLINE}, /* TAG_TR */
64     {"td",      HTML_CLRLINE}, /* TAG_TD */
65     {"li",      HTML_CLRLINE}, /* TAG_LI */
66     {"ul",      HTML_CLRLINE}, /* TAG_UL */
67     {"ol",      HTML_CLRLINE}, /* TAG_OL */
68     {"dl",      HTML_CLRLINE}, /* TAG_DL */
69     {"dt",      HTML_CLRLINE}, /* TAG_DT */
70     {"dd",      HTML_CLRLINE}, /* TAG_DD */
71     {"blockquote", HTML_CLRLINE}, /* TAG_BLOCKQUOTE */
72     {"p",       HTML_CLRLINE | HTML_NOSTACK | HTML_AUTOCLOSE}, /* TAG_P
73     {"pre",     HTML_CLRLINE}, /* TAG_PRE */
74     {"b",       0}, /* TAG_B */
75     {"i",       0}, /* TAG_I */
76     {"code",    0}, /* TAG_CODE */
77     {"small",   0}, /* TAG_SMALL */
78 };
79
80 static const char    *const htmlattrs[ATTR_MAX] = {
81     "http-equiv", /* ATTR_HTTP_EQUIV */
82     "content", /* ATTR_CONTENT */
83     "name", /* ATTR_NAME */
84     "rel", /* ATTR_REL */
85     "href", /* ATTR_HREF */
86     "type", /* ATTR_TYPE */
87     "media", /* ATTR_MEDIA */
88     "class", /* ATTR_CLASS */
89     "style", /* ATTR_STYLE */
90     "width", /* ATTR_WIDTH */
91     "id", /* ATTR_ID */
92     "summary", /* ATTR_SUMMARY */
93     "align", /* ATTR_ALIGN */
94     "colspan", /* ATTR_COLSPAN */
95 };
96
97 static const char    *const roffscales[SCALE_MAX] = {
98     "cm", /* SCALE_CM */
99     "in", /* SCALE_IN */
100    "pc", /* SCALE_PC */
101    "pt", /* SCALE_PT */
102    "em", /* SCALE_EM */
103    "em", /* SCALE_MM */
104    "ex", /* SCALE_EN */
105    "ex", /* SCALE_BU */
106    "em", /* SCALE_VS */
107    "ex", /* SCALE_FS */
108 };
109
110 static void          bufncat(struct html *, const char *, size_t);
111 static void          print_ctag(struct html *, enum htmltag);
112 static int           print_encode(struct html *, const char *, int);
113 static void          print_metaf(struct html *, enum mandoc_esc);
114 static void          print_attr(struct html *, const char *, const char *);
115 static void          *ml_alloc(char *, enum htmltype);
116
117 static void *
118 ml_alloc(char *outopts, enum htmltype type)
119 {
120     struct html      *h;
121     const char      *toks[5];
122     char             *v;

```

```

124     toks[0] = "style";
125     toks[1] = "man";
126     toks[2] = "includes";
127     toks[3] = "fragment";
128     toks[4] = NULL;

130     h = mandoc_calloc(1, sizeof(struct html));

132     h->type = type;
133     h->tags.head = NULL;
134     h->syntab = mchars_alloc();

136     while (outopts && *outopts)
137         switch (getsubopt(&outopts, UNCONST(toks), &v)) {
138             case (0):
139                 h->style = v;
140                 break;
141             case (1):
142                 h->base_man = v;
143                 break;
144             case (2):
145                 h->base_includes = v;
146                 break;
147             case (3):
148                 h->oflags |= HTML_FRAGMENT;
149                 break;
150             default:
151                 break;
152         }

154     return(h);
155 }

157 void *
158 html_alloc(char *outopts)
159 {
161     return(ml_alloc(outopts, HTML_HTML_4_01_STRICT));
162 }

165 void *
166 xhtml_alloc(char *outopts)
167 {
169     return(ml_alloc(outopts, HTML_XHTML_1_0_STRICT));
170 }

173 void
174 html_free(void *p)
175 {
176     struct tag     *tag;
177     struct html    *h;

179     h = (struct html *)p;

181     while ((tag = h->tags.head) != NULL) {
182         h->tags.head = tag->next;
183         free(tag);
184     }

185     if (h->syntab)
186         mchars_free(h->syntab);
187 }

```

```

189     free(h);
190 }

193 void
194 print_gen_head(struct html *h)
195 {
196     struct htmlpair tag[4];

198     tag[0].key = ATTR_HTTPEQUIV;
199     tag[0].val = "Content-Type";
200     tag[1].key = ATTR_CONTENT;
201     tag[1].val = "text/html; charset=utf-8";
202     print_otag(h, TAG_META, 2, tag);

204     tag[0].key = ATTR_NAME;
205     tag[0].val = "resource-type";
206     tag[1].key = ATTR_CONTENT;
207     tag[1].val = "document";
208     print_otag(h, TAG_META, 2, tag);

210     if (h->style) {
211         tag[0].key = ATTR_REL;
212         tag[0].val = "stylesheet";
213         tag[1].key = ATTR_HREF;
214         tag[1].val = h->style;
215         tag[2].key = ATTR_TYPE;
216         tag[2].val = "text/css";
217         tag[3].key = ATTR_MEDIA;
218         tag[3].val = "all";
219         print_otag(h, TAG_LINK, 4, tag);
220     }
221 }

223 static void
224 print_metaf(struct html *h, enum mandoc_esc deco)
225 {
226     enum htmlfont    font;

228     switch (deco) {
229     case (ESCAPE_FONTPREV):
230         font = h->metal;
231         break;
232     case (ESCAPE_FONTITALIC):
233         font = HTMLFONT_ITALIC;
234         break;
235     case (ESCAPE_FONTBOLD):
236         font = HTMLFONT_BOLD;
237         break;
238     case (ESCAPE_FONT):
239         /* FALLTHROUGH */
240     case (ESCAPE_FONTRoman):
241         font = HTMLFONT_NONE;
242         break;
243     default:
244         abort();
245         /* NOTREACHED */
246     }

248     if (h->metaf) {
249         print_tagq(h, h->metaf);
250         h->metaf = NULL;
251     }

253     h->metal = h->metac;
254     h->metac = font;

```



```

256     if (HTMLFONT_NONE != font)
257         h->metaf = HTMLFONT_BOLD == font ?
258             print_otag(h, TAG_B, 0, NULL) :
259             print_otag(h, TAG_I, 0, NULL);
260 }

262 int
263 html_strlen(const char *cp)
264 {
265     int          ssize, sz;
266     const char   *seq, *p;

268     /*
269      * Account for escaped sequences within string length
270      * calculations. This follows the logic in term_strlen() as we
271      * must calculate the width of produced strings.
272      * Assume that characters are always width of "1". This is
273      * hacky, but it gets the job done for approximation of widths.
274      */

276     sz = 0;
277     while (NULL != (p = strchr(cp, '\\'))) {
278         sz += (int)(p - cp);
279         ++cp;
280         switch (mandoc_escape(&cp, &seq, &ssize)) {
281             case (ESCAPE_ERROR):
282                 return(sz);
283             case (ESCAPE_UNICODE):
284                 /* FALLTHROUGH */
285             case (ESCAPE_NUMBERED):
286                 /* FALLTHROUGH */
287             case (ESCAPE_SPECIAL):
288                 sz++;
289                 break;
290             default:
291                 break;
292         }
293     }

295     assert(sz >= 0);
296     return(sz + strlen(cp));
297 }

299 static int
300 print_encode(struct html *h, const char *p, int norecure)
301 {
302     size_t      sz;
303     int         c, len, nospace;
304     const char  *seq;
305     enum mandoc_esc esc;
306     static const char rejs[6] = { '\\', '<', '>', '&', ASCII_HYPH, '\\0' };

308     nospace = 0;

310     while ('\\0' != *p) {
311         sz = strcspn(p, rejs);
312
313         fwrite(p, 1, sz, stdout);
314         p += (int)sz;

316         if ('\\0' == *p)
317             break;

319         switch (*p++) {
320             case ('<'):

```

```

321             printf("&lt;");
322             continue;
323         case ('>'):
324             printf("&gt;");
325             continue;
326         case ('&'):
327             printf("&amp;");
328             continue;
329         case (ASCII_HYPH):
330             putchar('-');
331             continue;
332         default:
333             break;
334     }

336     esc = mandoc_escape(&p, &seq, &len);
337     if (ESCAPE_ERROR == esc)
338         break;

340     switch (esc) {
341     case (ESCAPE_UNICODE):
342         /* Skip passed "u" header. */
343         c = mchars_num2uc(seq + 1, len - 1);
344         if ('\\0' != c)
345             printf("#%x", c);
346         break;
347     case (ESCAPE_NUMBERED):
348         c = mchars_num2char(seq, len);
349         if ('\\0' != c)
350             putchar(c);
351         break;
352     case (ESCAPE_SPECIAL):
353         c = mchars_spec2cp(h->symtab, seq, len);
354         if (c > 0)
355             printf("#%d", c);
356         else if (-1 == c && 1 == len)
357             putchar((int)*seq);
358         break;
359     case (ESCAPE_FONT):
360         /* FALLTHROUGH */
361     case (ESCAPE_FONTPREV):
362         /* FALLTHROUGH */
363     case (ESCAPE_FONTBOLD):
364         /* FALLTHROUGH */
365     case (ESCAPE_FONTITALIC):
366         /* FALLTHROUGH */
367     case (ESCAPE_FONTRoman):
368         if (norecure)
369             break;
370         print_metaf(h, esc);
371         break;
372     case (ESCAPE_NOSPACE):
373         if ('\\0' == *p)
374             nospace = 1;
375         break;
376     default:
377         break;
378     }

379 }

381     return(nospace);
382 }

385 static void
386 print_attr(struct html *h, const char *key, const char *val)

```

```

387 {
388     printf(" %s=\"", key);
389     (void)print_encode(h, val, 1);
390     putchar('\n');
391 }

394 struct tag *
395 print_otag(struct html *h, enum htmltag tag,
396            int sz, const struct htmlpair *p)
397 {
398     int            i;
399     struct tag     *t;

401     /* Push this tags onto the stack of open scopes. */

403     if ( ! (HTML_NOSTACK & htmltags[tag].flags)) {
404         t = mandoc_malloc(sizeof(struct tag));
405         t->tag = tag;
406         t->next = h->tags.head;
407         h->tags.head = t;
408     } else
409         t = NULL;

411     if ( ! (HTML_NOSPACE & h->flags))
412         if ( ! (HTML_CLRLINE & htmltags[tag].flags)) {
413             /* Manage keeps! */
414             if ( ! (HTML_KEEP & h->flags)) {
415                 if (HTML_PREKEEP & h->flags)
416                     h->flags |= HTML_KEEP;
417                 putchar(' ');
418             } else
419                 printf("#&#160;");
420         }

422     if ( ! (h->flags & HTML_NONOSPACE))
423         h->flags &= ~HTML_NOSPACE;
424     else
425         h->flags |= HTML_NOSPACE;

427     /* Print out the tag name and attributes. */

429     printf("<%s", htmltags[tag].name);
430     for (i = 0; i < sz; i++)
431         print_attr(h, htmlattrs[p[i].key], p[i].val);

433     /* Add non-overridable attributes. */

435     if (TAG_HTML == tag && HTML_XHTML_1_0_STRICT == h->type) {
436         print_attr(h, "xmlns", "http://www.w3.org/1999/xhtml");
437         print_attr(h, "xml:lang", "en");
438         print_attr(h, "lang", "en");
439     }

441     /* Accommodate for XML "well-formed" singleton escaping. */

443     if (HTML_AUTOCLOSE & htmltags[tag].flags)
444         switch (h->type) {
445             case (HTML_XHTML_1_0_STRICT):
446                 putchar('/');
447                 break;
448             default:
449                 break;
450         }

452     putchar('>');

```

```

454     h->flags |= HTML_NOSPACE;

456     if ((HTML_AUTOCLOSE | HTML_CLRLINE) & htmltags[tag].flags)
457         putchar('\n');

459     return(t);
460 }

463 static void
464 print_ctag(struct html *h, enum htmltag tag)
465 {
466     printf("</%s>", htmltags[tag].name);
467     if (HTML_CLRLINE & htmltags[tag].flags) {
468         h->flags |= HTML_NOSPACE;
469         putchar('\n');
470     }
471 }

474 void
475 print_gen_decls(struct html *h)
476 {
477     const char     *doctype;
478     const char     *dtd;
479     const char     *name;

481     switch (h->type) {
482     case (HTML_HTML_4_01_STRICT):
483         name = "HTML";
484         doctype = "-//W3C//DTD HTML 4.01//EN";
485         dtd = "http://www.w3.org/TR/html4/strict.dtd";
486         break;
487     default:
488         puts("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
489         name = "html";
490         doctype = "-//W3C//DTD XHTML 1.0 Strict//EN";
491         dtd = "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd";
492         break;
493     }

495     printf("<!DOCTYPE %s PUBLIC \"%s\" \"%s\">\n",
496           name, doctype, dtd);
497 }

499 void
500 print_text(struct html *h, const char *word)
501 {
503     if ( ! (HTML_NOSPACE & h->flags)) {
504         /* Manage keeps! */
505         if ( ! (HTML_KEEP & h->flags)) {
506             if (HTML_PREKEEP & h->flags)
507                 h->flags |= HTML_KEEP;
508             putchar(' ');
509         } else
510             printf("#&#160;");
511     }

513     assert(NULL == h->metaf);
514     if (HTMLFONT_NONE != h->metaf)
515         h->metaf = HTMLFONT_BOLD == h->metaf ?
516             print_otag(h, TAG_B, 0, NULL) :
517             print_otag(h, TAG_I, 0, NULL);

```

```

519     assert(word);
520     if ( ! print_encode(h, word, 0) ) {
521         if ( ! (h->flags & HTML_NONOSPACE) )
522             h->flags &= ~HTML_NOSPACE;
523     } else
524         h->flags |= HTML_NOSPACE;

526     if (h->metaf) {
527         print_tagq(h, h->metaf);
528         h->metaf = NULL;
529     }

531     h->flags &= ~HTML_IGNDELIM;
532 }

535 void
536 print_tagq(struct html *h, const struct tag *until)
537 {
538     struct tag    *tag;

540     while ((tag = h->tags.head) != NULL) {
541         /*
542          * Remember to close out and nullify the current
543          * meta-font and table, if applicable.
544          */
545         if (tag == h->metaf)
546             h->metaf = NULL;
547         if (tag == h->tblt)
548             h->tblt = NULL;
549         print_ctag(h, tag->tag);
550         h->tags.head = tag->next;
551         free(tag);
552         if (until && tag == until)
553             return;
554     }
555 }

558 void
559 print_stagq(struct html *h, const struct tag *suntil)
560 {
561     struct tag    *tag;

563     while ((tag = h->tags.head) != NULL) {
564         if (suntil && tag == suntil)
565             return;
566         /*
567          * Remember to close out and nullify the current
568          * meta-font and table, if applicable.
569          */
570         if (tag == h->metaf)
571             h->metaf = NULL;
572         if (tag == h->tblt)
573             h->tblt = NULL;
574         print_ctag(h, tag->tag);
575         h->tags.head = tag->next;
576         free(tag);
577     }
578 }

580 void
581 bufinit(struct html *h)
582 {
584     h->buf[0] = '\0';

```

```

585         h->buflen = 0;
586     }

588 void
589 bufcat_style(struct html *h, const char *key, const char *val)
590 {
592     bufcat(h, key);
593     bufcat(h, ":");
594     bufcat(h, val);
595     bufcat(h, ";");
596 }

598 void
599 bufcat(struct html *h, const char *p)
600 {
602     h->buflen = strlcat(h->buf, p, BUFSIZ);
603     assert(h->buflen < BUFSIZ);
604 }

606 void
607 bufcat_fmt(struct html *h, const char *fmt, ...)
608 {
609     va_list        ap;

611     va_start(ap, fmt);
612     (void)vsnprintf(h->buf + (int)h->buflen,
613                    BUFSIZ - h->buflen - 1, fmt, ap);
614     va_end(ap);
615     h->buflen = strlen(h->buf);
616 }

618 static void
619 bufncat(struct html *h, const char *p, size_t sz)
620 {
622     assert(h->buflen + sz + 1 < BUFSIZ);
623     strncat(h->buf, p, sz);
624     h->buflen += sz;
625 }

627 void
628 buffmt_includes(struct html *h, const char *name)
629 {
630     const char    *p, *pp;

632     pp = h->base_includes;
633
634     bufinit(h);
635     while (NULL != (p = strchr(pp, '%'))) {
636         bufncat(h, pp, (size_t)(p - pp));
637         switch (*(p + 1)) {
638             case('I'):
639                 bufcat(h, name);
640                 break;
641             default:
642                 bufncat(h, p, 2);
643                 break;
644         }
645         pp = p + 2;
646     }
647     if (pp)
648         bufcat(h, pp);
649 }

```

```
651 void
652 buffmt_man(struct html *h,
653             const char *name, const char *sec)
654 {
655     const char    *p, *pp;
656
657     pp = h->base_man;
658
659     bufinit(h);
660     while (NULL != (p = strchr(pp, '%'))) {
661         bufncat(h, pp, (size_t)(p - pp));
662         switch (*(p + 1)) {
663             case('S'):
664                 bufcat(h, sec ? sec : "1");
665                 break;
666             case('N'):
667                 bufcat_fmt(h, name);
668                 break;
669             default:
670                 bufncat(h, p, 2);
671                 break;
672         }
673         pp = p + 2;
674     }
675     if (pp)
676         bufcat(h, pp);
677 }
678
679 void
680 bufcat_su(struct html *h, const char *p, const struct roffsu *su)
681 {
682     double        v;
683
684     v = su->scale;
685     if (SCALE_MM == su->unit && 0.0 == (v /= 100.0))
686         v = 1.0;
687
688     bufcat_fmt(h, "%s: %.2f%s;", p, v, roffscales[su->unit]);
689 }
690
691 void
692 bufcat_id(struct html *h, const char *src)
693 {
694
695     /* Cf. <http://www.w3.org/TR/html4/types.html#h-6.2>. */
696
697     while ('\0' != *src)
698         bufcat_fmt(h, "%.2x", *src++);
699 }
```

```
*****
```

```
4165 Sat Jul 19 19:24:29 2014
```

```
new/usr/src/cmd/mandoc/html.h
```

```
5025 import and use mandoc
```

```
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
```

```
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
```

```
Reviewed by: Robert Mustacchi <rm@joyent.com>
```

```
Reviewed by: Albert Lee <trisk@nexenta.com>
```

```
Approved by: TBD
```

```
*****
```

```
1 /* $Id: html.h,v 1.47 2011/10/05 21:35:17 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef HTML_H
18 #define HTML_H
19
20 _BEGIN_DECLS
21
22 enum htmltag {
23     TAG_HTML,
24     TAG_HEAD,
25     TAG_BODY,
26     TAG_META,
27     TAG_TITLE,
28     TAG_DIV,
29     TAG_H1,
30     TAG_H2,
31     TAG_SPAN,
32     TAG_LINK,
33     TAG_BR,
34     TAG_A,
35     TAG_TABLE,
36     TAG_TBODY,
37     TAG_COL,
38     TAG_TR,
39     TAG_TD,
40     TAG_LI,
41     TAG_UL,
42     TAG_OL,
43     TAG_DL,
44     TAG_DT,
45     TAG_DD,
46     TAG_BLOCKQUOTE,
47     TAG_P,
48     TAG_PRE,
49     TAG_B,
50     TAG_I,
51     TAG_CODE,
52     TAG_SMALL,
53     TAG_MAX
54 };
55
56 enum htmlattr {
```

```
57     ATTR_HTTP EQUIV,
58     ATTR_CONTENT,
59     ATTR_NAME,
60     ATTR_REL,
61     ATTR_HREF,
62     ATTR_TYPE,
63     ATTR_MEDIA,
64     ATTR_CLASS,
65     ATTR_STYLE,
66     ATTR_WIDTH,
67     ATTR_ID,
68     ATTR_SUMMARY,
69     ATTR_ALIGN,
70     ATTR_COLSPAN,
71     ATTR_MAX
72 };
73
74 enum htmlfont {
75     HTMLFONT_NONE = 0,
76     HTMLFONT_BOLD,
77     HTMLFONT_ITALIC,
78     HTMLFONT_MAX
79 };
80
81 struct tag {
82     struct tag *next;
83     enum htmltag tag;
84 };
85
86 struct tagq {
87     struct tag *head;
88 };
89
90 struct htmlpair {
91     enum htmlattr key;
92     const char *val;
93 };
94
95 #define PAIR_INIT(p, t, v) \
96     do { \
97         (p)->key = (t); \
98         (p)->val = (v); \
99     } while (/* CONSTCOND */ 0)
100
101 #define PAIR_ID_INIT(p, v) PAIR_INIT(p, ATTR_ID, v)
102 #define PAIR_CLASS_INIT(p, v) PAIR_INIT(p, ATTR_CLASS, v)
103 #define PAIR_HREF_INIT(p, v) PAIR_INIT(p, ATTR_HREF, v)
104 #define PAIR_STYLE_INIT(p, h) PAIR_INIT(p, ATTR_STYLE, (h)->buf)
105 #define PAIR_SUMMARY_INIT(p, v) PAIR_INIT(p, ATTR_SUMMARY, v)
106
107 enum htmltype {
108     HTML_HTML_4_01_STRICT,
109     HTML_XHTML_1_0_STRICT
110 };
111
112 struct html {
113     int flags;
114 #define HTML_NOSPACE (1 << 0) /* suppress next space */
115 #define HTML_IGNORE_DELIM (1 << 1)
116 #define HTML_KEEP (1 << 2)
117 #define HTML_PREKEEP (1 << 3)
118 #define HTML_NONOSPACE (1 << 4) /* never add spaces */
119 #define HTML_LITERAL (1 << 5) /* literal (e.g., <PRE>) context */
120     struct tagq tags; /* stack of open tags */
121     struct rofftbl tbl; /* current table */
122     struct tag *tblt; /* current open table scope */

```

```
123     struct mchars    *symtab; /* character-escapes */
124     char             *base_man; /* base for manpage href */
125     char             *base_includes; /* base for include href */
126     char             *style; /* style-sheet URI */
127     char             bufc[BUFSIZ]; /* see bufcat and friends */
128     size_t           buflen;
129     struct tag        *metaf; /* current open font scope */
130     enum htmlfont     metal; /* last used font */
131     enum htmlfont     metac; /* current font mode */
132     enum htmltype     type; /* output media type */
133     int              oflags; /* output options */
134 #define HTML_FRAGMENT (1 << 0) /* don't emit HTML/HEAD/BODY */
135 };

137 void             print_gen_decls(struct html *);
138 void             print_gen_head(struct html *);
139 struct tag        *print_otag(struct html *, enum htmltag,
140                               int, const struct htmlpair *);
141 void             print_tagq(struct html *, const struct tag *);
142 void             print_stagq(struct html *, const struct tag *);
143 void             print_text(struct html *, const char *);
144 void             print_tblclose(struct html *);
145 void             print_tbl(struct html *, const struct tbl_span *);
146 void             print_eqn(struct html *, const struct eqn *);

148 void             bufcat_fmt(struct html *, const char *, ...);
149 void             bufcat(struct html *, const char *);
150 void             bufcat_id(struct html *, const char *);
151 void             bufcat_style(struct html *,
152                               const char *, const char *);
153 void             bufcat_su(struct html *, const char *,
154                               const struct roffsu *);
155 void             buffinit(struct html *);
156 void             buffmt_man(struct html *,
157                               const char *, const char *);
158 void             buffmt_includes(struct html *, const char *);

160 int             html_strlen(const char *);

162 __END_DECLS

164 #endif /*!HTML_H*/
```

new/usr/src/cmd/mandoc/lib.c

1

1155 Sat Jul 19 19:24:29 2014

new/usr/src/cmd/mandoc/lib.c

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /*      $Id: lib.c,v 1.9 2011/03/22 14:33:05 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <stdlib.h>
22 #include <string.h>
23 #include <time.h>
24
25 #include "mdoc.h"
26 #include "mandoc.h"
27 #include "libmdoc.h"
28
29 #define LINE(x, y) \
30     if (0 == strcmp(p, x)) return(y);
31
32 const char *
33 mdoc_a2lib(const char *p)
34 {
35
36     #include "lib.in"
37
38     return(NULL);
39 }
```

new/usr/src/cmd/mandoc/lib.in

1

497 Sat Jul 19 19:24:29 2014

new/usr/src/cmd/mandoc/lib.in

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

1 /*

2 * This file and its contents are supplied under the terms of the
3 * Common Development and Distribution License ("CDDL"), version 1.0.
4 * You may only use this file in accordance with the terms of version
5 * 1.0 of the CDDL.

6 *

7 * A full copy of the text of the CDDL should have accompanied this
8 * source. A copy of the CDDL is also available via the Internet at
9 * <http://www.illumos.org/license/CDDL>.

10 */

12 /*

13 * Copyright 2012 Nexenta Systems, Inc. All rights reserved.

14 */

16 /*

17 * TBD

18 */


```

*****
3102 Sat Jul 19 19:24:29 2014
new/usr/src/cmd/mandoc/libman.h
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: libman.h,v 1.55 2011/11/07 01:24:40 schwarze Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef LIBMAN_H
18 #define LIBMAN_H
19
20 enum man_next {
21     MAN_NEXT_SIBLING = 0,
22     MAN_NEXT_CHILD
23 };
24
25 struct man {
26     struct mparse *parse; /* parse pointer */
27     int flags; /* parse flags */
28 #define MAN_HALT (1 << 0) /* badness happened: die */
29 #define MAN_ELINE (1 << 1) /* Next-line element scope. */
30 #define MAN_BLINE (1 << 2) /* Next-line block scope. */
31 #define MAN_ILINE (1 << 3) /* Ignored in next-line scope. */
32 #define MAN_LITERAL (1 << 4) /* Literal input. */
33 #define MAN_BPLINE (1 << 5)
34 #define MAN_NEWLINE (1 << 6) /* first macro/text in a line */
35     enum man_next next; /* where to put the next node */
36     struct man_node *last; /* the last parsed node */
37     struct man_node *first; /* the first parsed node */
38     struct man_meta meta; /* document meta-data */
39     struct roff *roff;
40 };
41
42 #define MACRO_PROT_ARGS struct man *m, \
43     enum mant tok, \
44     int line, \
45     int ppos, \
46     int *pos, \
47     char *buf
48
49 struct man_macro {
50     int (*fp)(MACRO_PROT_ARGS);
51     int flags;
52 #define MAN_SCOPED (1 << 0)
53 #define MAN_EXPLICIT (1 << 1) /* See blk_imp(). */
54 #define MAN_FSCOPED (1 << 2) /* See blk_imp(). */
55 #define MAN_NSOPED (1 << 3) /* See in_line_eoln(). */
56 #define MAN_NOCLOSE (1 << 4) /* See blk_exp(). */

```

```

57 #define MAN_BSCOPE (1 << 5) /* Break BLINE scope. */
58 };
59
60 extern const struct man_macro *const man_macros;
61
62 __BEGIN_DECLS
63
64 #define man_pmsg(m, l, p, t) \
65     mandoc_msg((t), (m)->parse, (l), (p), NULL)
66 #define man_nmsg(m, n, t) \
67     mandoc_msg((t), (m)->parse, (n)->line, (n)->pos, NULL)
68 int man_word_alloc(struct man *, int, int, const char *);
69 int man_block_alloc(struct man *, int, int, enum mant);
70 int man_head_alloc(struct man *, int, int, enum mant);
71 int man_tail_alloc(struct man *, int, int, enum mant);
72 int man_body_alloc(struct man *, int, int, enum mant);
73 int man_elem_alloc(struct man *, int, int, enum mant);
74 void man_node_delete(struct man *, struct man_node *);
75 void man_hash_init(void);
76 enum mant man_hash_find(const char *);
77 int man_macroend(struct man *);
78 int man_valid_post(struct man *);
79 int man_valid_pre(struct man *, struct man_node *);
80 int man_unscope(struct man *,
81     const struct man_node *, enum mandocerr);
82
83 __END_DECLS
84
85 #endif /* !LIBMAN_H */

```

new/usr/src/cmd/mandoc/libmandoc.h

1

3343 Sat Jul 19 19:24:29 2014

new/usr/src/cmd/mandoc/libmandoc.h

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /* $Id: libmandoc.h,v 1.29 2011/12/02 01:37:14 schwarze Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef LIBMANDOC_H
18 #define LIBMANDOC_H
19
20 enum rofferr {
21     ROFF_CONT, /* continue processing line */
22     ROFF_RERUN, /* re-run roff interpreter with offset */
23     ROFF_APPEND, /* re-run main parser, appending next line */
24     ROFF_REPARSE, /* re-run main parser on the result */
25     ROFF_SO, /* include another file */
26     ROFF_IGN, /* ignore current line */
27     ROFF_TBL, /* a table row was successfully parsed */
28     ROFF_EQN, /* an equation was successfully parsed */
29     ROFF_ERR /* badness: puke and stop */
30 };
31
32 enum regs {
33     REG_nS = 0, /* nS register */
34     REG_MAX
35 };
36
37 __BEGIN_DECLS
38
39 struct roff;
40 struct mdoc;
41 struct man;
42
43 void mandoc_msg(enum mandocerr, struct mparse *,
44                 int, int, const char *);
45 void mandoc_vmsg(enum mandocerr, struct mparse *,
46                 int, int, const char *, ...);
47 char *mandoc_getarg(struct mparse *, char **, int, int *);
48 char *mandoc_normdate(struct mparse *, char *, int, int);
49 int mandoc_eos(const char *, size_t, int);
50 int mandoc_getcontrol(const char *, int *);
51 int mandoc_strntoi(const char *, size_t, int);
52 const char *mandoc_a2msec(const char *);
53
54 void mdoc_free(struct mdoc *);
55 struct mdoc *mdoc_alloc(struct roff *, struct mparse *);
56 void mdoc_reset(struct mdoc *);
```

new/usr/src/cmd/mandoc/libmandoc.h

2

```
57 int mdoc_parseln(struct mdoc *, int, char *, int);
58 int mdoc_endparse(struct mdoc *);
59 int mdoc_addspan(struct mdoc *, const struct tbl_span *);
60 int mdoc_addeqn(struct mdoc *, const struct eqn *);
61
62 void man_free(struct man *);
63 struct man *man_alloc(struct roff *, struct mparse *);
64 void man_reset(struct man *);
65 int man_parseln(struct man *, int, char *, int);
66 int man_endparse(struct man *);
67 int man_addspan(struct man *, const struct tbl_span *);
68 int man_addeqn(struct man *, const struct eqn *);
69
70 void roff_free(struct roff *);
71 struct roff *roff_alloc(struct mparse *);
72 void roff_reset(struct roff *);
73 enum rofferr roff_parseln(struct roff *, int,
74                          char **, size_t *, int, int *);
75 void roff_endparse(struct roff *);
76 int roff_regisset(const struct roff *, enum regs);
77 unsigned int roff_regget(const struct roff *, enum regs);
78 void roff_regunset(struct roff *, enum regs);
79 char *roff_strdup(const struct roff *, const char *);
80 #if 0
81 char *roff_eqndelim(const struct roff *);
82 void roff_openeqn(struct roff *, const char *,
83                  int, int, const char *);
84 int roff_closeqn(struct roff *);
85 #endif
86
87 const struct tbl_span *roff_span(const struct roff *);
88 const struct eqn *roff_eqn(const struct roff *);
89
90 __END_DECLS
91
92 #endif /* !LIBMANDOC_H */
```

```

*****
4877 Sat Jul 19 19:24:29 2014
new/usr/src/cmd/mandoc/libmdoc.h
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: libmdoc.h,v 1.78 2011/12/02 01:37:14 schwarze Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef LIBMDOC_H
18 #define LIBMDOC_H
19
20 enum mdoc_next {
21     MDOC_NEXT_SIBLING = 0,
22     MDOC_NEXT_CHILD
23 };
24
25 struct mdoc {
26     struct mparse *parse; /* parse pointer */
27     int flags; /* parse flags */
28 #define MDOC_HALT (1 << 0) /* error in parse: halt */
29 #define MDOC_LITERAL (1 << 1) /* in a literal scope */
30 #define MDOC_PBODY (1 << 2) /* in the document body */
31 #define MDOC_NEWLINE (1 << 3) /* first macro/text in a line */
32 #define MDOC_PHRASELIT (1 << 4) /* literal within a partial phrase */
33 #define MDOC_PPHRASE (1 << 5) /* within a partial phrase */
34 #define MDOC_FREECOL (1 << 6) /* 'It' invocation should close */
35 #define MDOC_SYNOPSIS (1 << 7) /* SYNOPSIS-style formatting */
36     enum mdoc_next next; /* where to put the next node */
37     struct mdoc_node *last; /* the last node parsed */
38     struct mdoc_node *first; /* the first node parsed */
39     struct mdoc_meta meta; /* document meta-data */
40     enum mdoc_sec lastnamed;
41     enum mdoc_sec lastsec;
42     struct roff *roff;
43 };
44
45 #define MACRO_PROT_ARGS struct mdoc *m, \
46     enum mdoct tok, \
47     int line, \
48     int ppos, \
49     int *pos, \
50     char *buf
51
52 struct mdoc_macro {
53     int (*fp)(MACRO_PROT_ARGS);
54     int flags;
55 #define MDOC_CALLABLE (1 << 0)
56 #define MDOC_PARSED (1 << 1)

```

```

57 #define MDOC_EXPLICIT (1 << 2)
58 #define MDOC_PROLOGUE (1 << 3)
59 #define MDOC_IGNDELIM (1 << 4)
60 /* Reserved words in arguments treated as text. */
61 };
62
63 enum margserr {
64     ARGS_ERROR,
65     ARGS_EOLN, /* end-of-line */
66     ARGS_WORD, /* normal word */
67     ARGS_PUNCT, /* series of punctuation */
68     ARGS_QWORD, /* quoted word */
69     ARGS_PHRASE, /* Ta'd phrase (-column) */
70     ARGS_PPHRASE, /* tabbed phrase (-column) */
71     ARGS_PEND /* last phrase (-column) */
72 };
73
74 enum margverr {
75     ARGV_ERROR,
76     ARGV_EOLN, /* end of line */
77     ARGV_ARG, /* valid argument */
78     ARGV_WORD /* normal word (or bad argument---same thing) */
79 };
80
81 /*
82  * A punctuation delimiter is opening, closing, or "middle mark"
83  * punctuation. These govern spacing.
84  * Opening punctuation (e.g., the opening parenthesis) suppresses the
85  * following space; closing punctuation (e.g., the closing parenthesis)
86  * suppresses the leading space; middle punctuation (e.g., the vertical
87  * bar) can do either. The middle punctuation delimiter bends the rules
88  * depending on usage.
89 */
90 enum mdelim {
91     DELIM_NONE = 0,
92     DELIM_OPEN,
93     DELIM_MIDDLE,
94     DELIM_CLOSE,
95     DELIM_MAX
96 };
97
98 extern const struct mdoc_macro *const mdoc_macros;
99
100 __BEGIN_DECLS
101
102 #define mdoc_pmsg(m, l, p, t) \
103     mandoc_msg((t), (m)->parse, (l), (p), NULL)
104 #define mdoc_rmsg(m, n, t) \
105     mandoc_msg((t), (m)->parse, (n)->line, (n)->pos, NULL)
106 int mdoc_macro(MACRO_PROT_ARGS);
107 int mdoc_word_alloc(struct mdoc *,
108     int, int, const char *);
109 int mdoc_elem_alloc(struct mdoc *, int, int,
110     enum mdoct, struct mdoc_arg *);
111 int mdoc_block_alloc(struct mdoc *, int, int,
112     enum mdoct, struct mdoc_arg *);
113 int mdoc_head_alloc(struct mdoc *, int, int, enum mdoct);
114 int mdoc_tail_alloc(struct mdoc *, int, int, enum mdoct);
115 int mdoc_body_alloc(struct mdoc *, int, int, enum mdoct);
116 int mdoc_endbody_alloc(struct mdoc *m, int line, int pos,
117     enum mdoct tok, struct mdoc_node *body,
118     enum mdoc_endbody end);
119 void mdoc_node_delete(struct mdoc *, struct mdoc_node *);
120 void mdoc_hash_init(void);
121 enum mdoct mdoc_hash_find(const char *);
122 const char *mdoc_a2att(const char *);

```

```
123 const char      *mdoc_a2lib(const char *);
124 const char      *mdoc_a2st(const char *);
125 const char      *mdoc_a2arch(const char *);
126 const char      *mdoc_a2vol(const char *);
127 int              mdoc_valid_pre(struct mdoc *, struct mdoc_node *);
128 int              mdoc_valid_post(struct mdoc *);
129 enum margverr    mdoc_argv(struct mdoc *, int, enum mdoct,
130                          struct mdoc_arg **, int *, char *);
131 void             mdoc_argv_free(struct mdoc_arg *);
132 enum margserr    mdoc_args(struct mdoc *, int,
133                          int *, char *, enum mdoct, char **);
134 enum margserr    mdoc_zargs(struct mdoc *, int,
135                          int *, char *, char **);
136 int              mdoc_macroend(struct mdoc *);
137 enum mdelim      mdoc_isdelim(const char *);

139 __END_DECLS

141 #endif /*!LIBMDOC_H*/
```

```
*****
```

```
2642 Sat Jul 19 19:24:29 2014
new/usr/src/cmd/mandoc/libroff.h
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
```

```
1 /* $Id: libroff.h,v 1.27 2011/07/25 15:37:00 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef LIBROFF_H
18 #define LIBROFF_H
19
20 __BEGIN_DECLS
21
22 enum tbl_part {
23     TBL_PART_OPTS, /* in options (first line) */
24     TBL_PART_LAYOUT, /* describing layout */
25     TBL_PART_DATA, /* creating data rows */
26     TBL_PART_CDATA /* continue previous row */
27 };
28
29 struct tbl_node {
30     struct mparse *parse; /* parse point */
31     int pos; /* invocation column */
32     int line; /* invocation line */
33     enum tbl_part part;
34     struct tbl opts;
35     struct tbl_row *first_row;
36     struct tbl_row *last_row;
37     struct tbl_span *first_span;
38     struct tbl_span *current_span;
39     struct tbl_span *last_span;
40     struct tbl_head *first_head;
41     struct tbl_head *last_head;
42     struct tbl_node *next;
43 };
44
45 struct eqn_node {
46     struct eqn_def *defs;
47     size_t defsz;
48     char *data;
49     size_t rew;
50     size_t cur;
51     size_t sz;
52     int gsize;
53     struct eqn eqn;
54     struct mparse *parse;
55     struct eqn_node *next;
56 };
```

```
58 struct eqn_def {
59     char *key;
60     size_t keysz;
61     char *val;
62     size_t valsz;
63 };
64
65 struct tbl_node *tbl_alloc(int, int, struct mparse *);
66 void tbl_restart(int, int, struct tbl_node *);
67 void tbl_free(struct tbl_node *);
68 void tbl_reset(struct tbl_node *);
69 enum rofferr tbl_read(struct tbl_node *, int, const char *, int);
70 int tbl_option(struct tbl_node *, int, const char *);
71 int tbl_layout(struct tbl_node *, int, const char *);
72 int tbl_data(struct tbl_node *, int, const char *);
73 int tbl_cdata(struct tbl_node *, int, const char *);
74 const struct tbl_span *tbl_span(struct tbl_node *);
75 void tbl_end(struct tbl_node **);
76 struct eqn_node *eqn_alloc(const char *, int, int, struct mparse *);
77 enum rofferr eqn_end(struct eqn_node **);
78 void eqn_free(struct eqn_node *);
79 enum rofferr eqn_read(struct eqn_node **, int,
80     const char *, int, int *);
81
82 __END_DECLS
83
84 #endif /*LIBROFF_H*/
```

```

*****
8872 Sat Jul 19 19:24:30 2014
new/usr/src/cmd/mandoc/main.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: main.c,v 1.165 2011/10/06 22:29:12 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2010, 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <assert.h>
23 #include <stdio.h>
24 #include <stdint.h>
25 #include <stdlib.h>
26 #include <string.h>
27 #include <unistd.h>
28
29 #include "mandoc.h"
30 #include "main.h"
31 #include "mdoc.h"
32 #include "man.h"
33
34 #if !defined(__GNUC__) || (__GNUC__ < 2)
35 # if !defined(lint)
36 #  define __attribute__(x)
37 # endif
38 #endif /* !defined(__GNUC__) || (__GNUC__ < 2) */
39
40 typedef void      (*out_mdoc)(void *, const struct mdoc *);
41 typedef void      (*out_man)(void *, const struct man *);
42 typedef void      (*out_free)(void *);
43
44 enum      outt {
45     OUTT_ASCII = 0, /* -Tascii */
46     OUTT_LOCALE, /* -Tlocale */
47     OUTT_UTF8, /* -Tutf8 */
48     OUTT_TREE, /* -Ttree */
49     OUTT_MAN, /* -Tman */
50     OUTT_HTML, /* -Thtml */
51     OUTT_XHTML, /* -Txhtml */
52     OUTT_LINT, /* -Tlint */
53     OUTT_PS, /* -Tps */
54     OUTT_PDF, /* -Tpdf */
55 };

```

```

57 struct curparse {
58     struct mparse *mp;
59     enum mandoclevel wlevel; /* ignore messages below this */
60     int wstop; /* stop after a file with a warning */
61     enum outt outtype; /* which output to use */
62     out_mdoc outmdoc; /* mdoc output ptr */
63     out_man outman; /* man output ptr */
64     out_free outfree; /* free output ptr */
65     void *outdata; /* data for output */
66     char outopts[BUFSIZ]; /* buf of output opts */
67 };
68
69 static int moptions(enum mparset *, char *);
70 static void mmsg(enum mandocerr, enum mandoclevel,
71                 const char *, int, int, const char *);
72 static void parse(struct curparse *, int,
73                  const char *, enum mandoclevel *);
74 static int toptions(struct curparse *, char *);
75 static void usage(void) __attribute__((noreturn));
76 static void version(void) __attribute__((noreturn));
77 static int woptions(struct curparse *, char *);
78
79 static const char *programe;
80
81 int
82 main(int argc, char *argv[])
83 {
84     int c;
85     struct curparse curp;
86     enum mparset type;
87     enum mandoclevel rc;
88
89     programe = strrchr(argv[0], '/');
90     if (programe == NULL)
91         programe = argv[0];
92     else
93         ++programe;
94
95     memset(&curp, 0, sizeof(struct curparse));
96
97     type = MPARSE_AUTO;
98     curp.outtype = OUTT_ASCII;
99     curp.wlevel = MANDOCLEVEL_FATAL;
100
101     /* LINTED */
102     while (-1 != (c = getopt(argc, argv, "m:O:T:VW:")))
103         switch (c) {
104             case ('m'):
105                 if (! moptions(&type, optarg))
106                     return((int)MANDOCLEVEL_BADARG);
107                 break;
108             case ('O'):
109                 (void)strlcat(curp.outopts, optarg, BUFSIZ);
110                 (void)strlcat(curp.outopts, ",", BUFSIZ);
111                 break;
112             case ('T'):
113                 if (! toptions(&curp, optarg))
114                     return((int)MANDOCLEVEL_BADARG);
115                 break;
116             case ('W'):
117                 if (! woptions(&curp, optarg))
118                     return((int)MANDOCLEVEL_BADARG);
119                 break;
120             case ('V'):
121                 version();
122                 /* NOTREACHED */

```

```

123     default:
124         usage();
125         /* NOTREACHED */
126     }
127
128     curp.mp = mparse_alloc(type, curp.wlevel, mmsg, &curp);
129
130     /*
131      * Conditionally start up the lookaside buffer before parsing.
132      */
133     if (OUTT_MAN == curp.outtype)
134         mparse_keep(curp.mp);
135
136     argc -= optind;
137     argv += optind;
138
139     rc = MANDOCLEVEL_OK;
140
141     if (NULL == *argv)
142         parse(&curp, STDIN_FILENO, "<stdin>", &rc);
143
144     while (*argv) {
145         parse(&curp, -1, *argv, &rc);
146         if (MANDOCLEVEL_OK != rc && curp.wstop)
147             break;
148         ++argv;
149     }
150
151     if (curp.outfree)
152         (*curp.outfree)(curp.outdata);
153     if (curp.mp)
154         mparse_free(curp.mp);
155
156     return((int)rc);
157 }
158
159 static void
160 version(void)
161 {
162
163     printf("%s %s\n", progname, VERSION);
164     exit((int)MANDOCLEVEL_OK);
165 }
166
167 static void
168 usage(void)
169 {
170
171     fprintf(stderr, "usage: %s "
172             "[-V] "
173             "[-foption] "
174             "[-mformat] "
175             "[-Option] "
176             "[-Toutput] "
177             "[-Wlevel] "
178             "[file...]\n",
179             progname);
180
181     exit((int)MANDOCLEVEL_BADARG);
182 }
183
184 static void
185 parse(struct curparse *curp, int fd,
186       const char *file, enum mandoclevel *level)
187 {
188     enum mandoclevel rc;

```

```

189     struct mdoc      *mdoc;
190     struct man       *man;
191
192     /* Begin by parsing the file itself. */
193
194     assert(file);
195     assert(fd >= -1);
196
197     rc = mparse_readfd(curp->mp, fd, file);
198
199     /* Stop immediately if the parse has failed. */
200
201     if (MANDOCLEVEL_FATAL <= rc)
202         goto cleanup;
203
204     /*
205      * With -Wstop and warnings or errors of at least the requested
206      * level, do not produce output.
207      */
208
209     if (MANDOCLEVEL_OK != rc && curp->wstop)
210         goto cleanup;
211
212     /* If unset, allocate output dev now (if applicable). */
213
214     if ( ! (curp->outman && curp->outmdoc) ) {
215         switch (curp->outtype) {
216             case (OUTT_XHTML):
217                 curp->outdata = xhtml_alloc(curp->outopts);
218                 curp->outfree = html_free;
219                 break;
220             case (OUTT_HTML):
221                 curp->outdata = html_alloc(curp->outopts);
222                 curp->outfree = html_free;
223                 break;
224             case (OUTT_UTF8):
225                 curp->outdata = utf8_alloc(curp->outopts);
226                 curp->outfree = ascii_free;
227                 break;
228             case (OUTT_LOCALE):
229                 curp->outdata = locale_alloc(curp->outopts);
230                 curp->outfree = ascii_free;
231                 break;
232             case (OUTT_ASCII):
233                 curp->outdata = ascii_alloc(curp->outopts);
234                 curp->outfree = ascii_free;
235                 break;
236             case (OUTT_PDF):
237                 curp->outdata = pdf_alloc(curp->outopts);
238                 curp->outfree = pspdf_free;
239                 break;
240             case (OUTT_PS):
241                 curp->outdata = ps_alloc(curp->outopts);
242                 curp->outfree = pspdf_free;
243                 break;
244             default:
245                 break;
246         }
247
248         switch (curp->outtype) {
249             case (OUTT_HTML):
250                 /* FALLTHROUGH */
251             case (OUTT_XHTML):
252                 curp->outman = html_man;
253                 curp->outmdoc = html_mdoc;
254                 break;

```

```

255     case (OUTT_TREE):
256         curp->outman = tree_man;
257         curp->outmdoc = tree_mdoc;
258         break;
259     case (OUTT_MAN):
260         curp->outmdoc = man_mdoc;
261         curp->outman = man_man;
262         break;
263     case (OUTT_PDF):
264         /* FALLTHROUGH */
265     case (OUTT_ASCII):
266         /* FALLTHROUGH */
267     case (OUTT_UTF8):
268         /* FALLTHROUGH */
269     case (OUTT_LOCALE):
270         /* FALLTHROUGH */
271     case (OUTT_PS):
272         curp->outman = terminal_man;
273         curp->outmdoc = terminal_mdoc;
274         break;
275     default:
276         break;
277 }
278
280 mparse_result(curp->mp, &mdoc, &man);
281
282 /* Execute the out device, if it exists. */
283
284 if (man && curp->outman)
285     (*curp->outman)(curp->outdata, man);
286 if (mdoc && curp->outmdoc)
287     (*curp->outmdoc)(curp->outdata, mdoc);
288
289 cleanup:
290
291 mparse_reset(curp->mp);
292
293 if (*level < rc)
294     *level = rc;
295 }
296
297 static int
298 moptions(enum mparset *tflags, char *arg)
299 {
300
301     if (0 == strcmp(arg, "doc"))
302         *tflags = MPARSE_MDOC;
303     else if (0 == strcmp(arg, "andoc"))
304         *tflags = MPARSE_AUTO;
305     else if (0 == strcmp(arg, "an"))
306         *tflags = MPARSE_MAN;
307     else {
308         fprintf(stderr, "%s: Bad argument\n", arg);
309         return(0);
310     }
311
312     return(1);
313 }
314
315 static int
316 toptions(struct curparse *curp, char *arg)
317 {
318
319     if (0 == strcmp(arg, "ascii"))
320         curp->outtype = OUTT_ASCII;

```

```

321     else if (0 == strcmp(arg, "lint")) {
322         curp->outtype = OUTT_LINT;
323         curp->wlevel = MANDOCLEVEL_WARNING;
324     } else if (0 == strcmp(arg, "tree"))
325         curp->outtype = OUTT_TREE;
326     else if (0 == strcmp(arg, "man"))
327         curp->outtype = OUTT_MAN;
328     else if (0 == strcmp(arg, "html"))
329         curp->outtype = OUTT_HTML;
330     else if (0 == strcmp(arg, "utf8"))
331         curp->outtype = OUTT_UTF8;
332     else if (0 == strcmp(arg, "locale"))
333         curp->outtype = OUTT_LOCALE;
334     else if (0 == strcmp(arg, "xhtml"))
335         curp->outtype = OUTT_XHTML;
336     else if (0 == strcmp(arg, "ps"))
337         curp->outtype = OUTT_PS;
338     else if (0 == strcmp(arg, "pdf"))
339         curp->outtype = OUTT_PDF;
340     else {
341         fprintf(stderr, "%s: Bad argument\n", arg);
342         return(0);
343     }
344
345     return(1);
346 }
347
348 static int
349 woptions(struct curparse *curp, char *arg)
350 {
351     char          *v, *o;
352     const char    *toks[6];
353
354     toks[0] = "stop";
355     toks[1] = "all";
356     toks[2] = "warning";
357     toks[3] = "error";
358     toks[4] = "fatal";
359     toks[5] = NULL;
360
361     while (*arg) {
362         o = arg;
363         switch (getsubopt(&arg, UNCONST(toks), &v)) {
364             case (0):
365                 curp->wstop = 1;
366                 break;
367             case (1):
368                 /* FALLTHROUGH */
369             case (2):
370                 curp->wlevel = MANDOCLEVEL_WARNING;
371                 break;
372             case (3):
373                 curp->wlevel = MANDOCLEVEL_ERROR;
374                 break;
375             case (4):
376                 curp->wlevel = MANDOCLEVEL_FATAL;
377                 break;
378             default:
379                 fprintf(stderr, "-W%s: Bad argument\n", o);
380                 return(0);
381         }
382     }
383
384     return(1);
385 }

```



```
387 static void
388 mmsg(enum mandocerr t, enum mandoclevel lvl,
389      const char *file, int line, int col, const char *msg)
390 {
391
392     fprintf(stderr, "%s:%d:%d: %s: %s",
393            file, line, col + 1,
394            mparse_strlevel(lvl),
395            mparse_strerror(t));
396
397     if (msg)
398         fprintf(stderr, ": %s", msg);
399
400     fputc('\n', stderr);
401 }
```

new/usr/src/cmd/mandoc/main.h

1

1974 Sat Jul 19 19:24:30 2014

new/usr/src/cmd/mandoc/main.h

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /*      $Id: main.h,v 1.15 2011/10/06 22:29:12 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef MAIN_H
18 #define MAIN_H
19
20 __BEGIN_DECLS
21
22 struct mdoc;
23 struct man;
24
25 #define UNCONST(a)      ((void*)(uintptr_t)(const void*)(a))
26
27
28 /*
29  * Definitions for main.c-visible output device functions, e.g., -Thtml
30  * and -Tascii. Note that ascii_alloc() is named as such in
31  * anticipation of latin1_alloc() and so on, all of which map into the
32  * terminal output routines with different character settings.
33  */
34
35 void      *html_alloc(char *);
36 void      *xhtml_alloc(char *);
37 void      html_mdoc(void *, const struct mdoc *);
38 void      html_man(void *, const struct man *);
39 void      html_free(void *);
40
41 void      tree_mdoc(void *, const struct mdoc *);
42 void      tree_man(void *, const struct man *);
43
44 void      man_mdoc(void *, const struct mdoc *);
45 void      man_man(void *, const struct man *);
46
47 void      *locale_alloc(char *);
48 void      *utf8_alloc(char *);
49 void      *ascii_alloc(char *);
50 void      *ascii_free(void *);
51
52 void      *pdf_alloc(char *);
53 void      *ps_alloc(char *);
54 void      pspdf_free(void *);
55
56 void      terminal_mdoc(void *, const struct mdoc *);
```

new/usr/src/cmd/mandoc/main.h

2

```
57 void      terminal_man(void *, const struct man *);
58
59 __END_DECLS
60
61 #endif /*!MAIN_H*/
```

```

*****
13782 Sat Jul 19 19:24:30 2014
new/usr/src/cmd/mandoc/man.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*
2  * $Id: man.c,v 1.115 2012/01/03 15:16:24 kristaps Exp $ */
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <sys/types.h>
22
23 #include <assert.h>
24 #include <stdarg.h>
25 #include <stdlib.h>
26 #include <stdio.h>
27 #include <string.h>
28
29 #include "man.h"
30 #include "mandoc.h"
31 #include "libman.h"
32 #include "libmandoc.h"
33
34 const char *const __man_macronames[MAN_MAX] = {
35     "br",      "TH",      "SH",      "SS",
36     "TP",      "LP",      "PP",      "P",
37     "IP",      "HP",      "SM",      "SB",
38     "BI",      "IB",      "BR",      "RB",
39     "R",       "B",       "I",       "IR",
40     "RI",      "na",      "sp",      "nf",
41     "fi",      "RE",      "RS",      "DT",
42     "UC",      "PD",      "AT",      "in",
43     "ft",      "OP",
44 };
45
46 const char * const *man_macronames = __man_macronames;
47
48 static struct man_node *man_node_alloc(struct man *, int, int,
49     enum man_type, enum mant);
50 static int man_node_append(struct man *,
51     struct man_node *);
52 static void man_node_free(struct man_node *);
53 static void man_node_unlink(struct man *,
54     struct man_node *);
55 static int man_ptext(struct man *, int, char *, int);
56 static int man_pmacro(struct man *, int, char *, int);

```

```

57 static void man_freel(struct man *);
58 static void man_alloc1(struct man *);
59 static int man_descope(struct man *, int, int);
60
61
62 const struct man_node *
63 man_node(const struct man *m)
64 {
65     assert( ! (MAN_HALT & m->flags));
66     return(m->first);
67 }
68
69
70
71 const struct man_meta *
72 man_meta(const struct man *m)
73 {
74     assert( ! (MAN_HALT & m->flags));
75     return(&m->meta);
76 }
77
78
79 void
80 man_reset(struct man *man)
81 {
82     man_freel(man);
83     man_alloc1(man);
84 }
85
86
87
88 void
89 man_free(struct man *man)
90 {
91     man_freel(man);
92     free(man);
93 }
94
95
96
97
98 struct man *
99 man_alloc(struct roff *roff, struct mparse *parse)
100 {
101     struct man *p;
102
103     p = mandoc_calloc(1, sizeof(struct man));
104
105     man_hash_init();
106     p->parse = parse;
107     p->roff = roff;
108
109     man_alloc1(p);
110     return(p);
111 }
112
113
114 int
115 man_endparse(struct man *m)
116 {
117     assert( ! (MAN_HALT & m->flags));
118     if (man_macroend(m))
119         return(1);
120     m->flags |= MAN_HALT;
121     return(0);

```

```

123 }

126 int
127 man_parseln(struct man *m, int ln, char *buf, int offs)
128 {
130     m->flags |= MAN_NEWLINE;
132     assert( ! (MAN_HALT & m->flags));
134     return (mandoc_getcontrol(buf, &offs) ?
135             man_pmacro(m, ln, buf, offs) :
136             man_ptext(m, ln, buf, offs));
137 }

140 static void
141 man_freel(struct man *man)
142 {
144     if (man->first)
145         man_node_delete(man, man->first);
146     if (man->meta.title)
147         free(man->meta.title);
148     if (man->meta.source)
149         free(man->meta.source);
150     if (man->meta.date)
151         free(man->meta.date);
152     if (man->meta.vol)
153         free(man->meta.vol);
154     if (man->meta.msec)
155         free(man->meta.msec);
156 }

159 static void
160 man_alloc1(struct man *m)
161 {
163     memset(&m->meta, 0, sizeof(struct man_meta));
164     m->flags = 0;
165     m->last = mandoc_calloc(1, sizeof(struct man_node));
166     m->first = m->last;
167     m->last->type = MAN_ROOT;
168     m->last->tok = MAN_MAX;
169     m->next = MAN_NEXT_CHILD;
170 }

173 static int
174 man_node_append(struct man *man, struct man_node *p)
175 {
177     assert(man->last);
178     assert(man->first);
179     assert(MAN_ROOT != p->type);

181     switch (man->next) {
182     case (MAN_NEXT_SIBLING):
183         man->last->next = p;
184         p->prev = man->last;
185         p->parent = man->last->parent;
186         break;
187     case (MAN_NEXT_CHILD):
188         man->last->child = p;

```

```

189         p->parent = man->last;
190         break;
191     default:
192         abort();
193         /* NOTREACHED */
194     }
195
196     assert(p->parent);
197     p->parent->nchild++;

199     if ( ! man_valid_pre(man, p))
200         return(0);

202     switch (p->type) {
203     case (MAN_HEAD):
204         assert(MAN_BLOCK == p->parent->type);
205         p->parent->head = p;
206         break;
207     case (MAN_TAIL):
208         assert(MAN_BLOCK == p->parent->type);
209         p->parent->tail = p;
210         break;
211     case (MAN_BODY):
212         assert(MAN_BLOCK == p->parent->type);
213         p->parent->body = p;
214         break;
215     default:
216         break;
217     }

219     man->last = p;

221     switch (p->type) {
222     case (MAN_TBL):
223         /* FALLTHROUGH */
224     case (MAN_TEXT):
225         if ( ! man_valid_post(man))
226             return(0);
227         break;
228     default:
229         break;
230     }

232     return(1);
233 }

236 static struct man_node *
237 man_node_alloc(struct man *m, int line, int pos,
238               enum man_type type, enum mant tok)
239 {
240     struct man_node *p;

242     p = mandoc_calloc(1, sizeof(struct man_node));
243     p->line = line;
244     p->pos = pos;
245     p->type = type;
246     p->tok = tok;

248     if (MAN_NEWLINE & m->flags)
249         p->flags |= MAN_LINE;
250     m->flags &= ~MAN_NEWLINE;
251     return(p);
252 }

```

```

255 int
256 man_elem_alloc(struct man *m, int line, int pos, enum mant tok)
257 {
258     struct man_node *p;

260     p = man_node_alloc(m, line, pos, MAN_ELEM, tok);
261     if ( ! man_node_append(m, p))
262         return(0);
263     m->next = MAN_NEXT_CHILD;
264     return(1);
265 }

268 int
269 man_tail_alloc(struct man *m, int line, int pos, enum mant tok)
270 {
271     struct man_node *p;

273     p = man_node_alloc(m, line, pos, MAN_TAIL, tok);
274     if ( ! man_node_append(m, p))
275         return(0);
276     m->next = MAN_NEXT_CHILD;
277     return(1);
278 }

281 int
282 man_head_alloc(struct man *m, int line, int pos, enum mant tok)
283 {
284     struct man_node *p;

286     p = man_node_alloc(m, line, pos, MAN_HEAD, tok);
287     if ( ! man_node_append(m, p))
288         return(0);
289     m->next = MAN_NEXT_CHILD;
290     return(1);
291 }

294 int
295 man_body_alloc(struct man *m, int line, int pos, enum mant tok)
296 {
297     struct man_node *p;

299     p = man_node_alloc(m, line, pos, MAN_BODY, tok);
300     if ( ! man_node_append(m, p))
301         return(0);
302     m->next = MAN_NEXT_CHILD;
303     return(1);
304 }

307 int
308 man_block_alloc(struct man *m, int line, int pos, enum mant tok)
309 {
310     struct man_node *p;

312     p = man_node_alloc(m, line, pos, MAN_BLOCK, tok);
313     if ( ! man_node_append(m, p))
314         return(0);
315     m->next = MAN_NEXT_CHILD;
316     return(1);
317 }

319 int
320 man_word_alloc(struct man *m, int line, int pos, const char *word)

```

```

321 {
322     struct man_node *n;

324     n = man_node_alloc(m, line, pos, MAN_TEXT, MAN_MAX);
325     n->string = roff_strdup(m->roff, word);

327     if ( ! man_node_append(m, n))
328         return(0);

330     m->next = MAN_NEXT_SIBLING;
331     return(1);
332 }

335 /*
336  * Free all of the resources held by a node. This does NOT unlink a
337  * node from its context; for that, see man_node_unlink().
338  */
339 static void
340 man_node_free(struct man_node *p)
341 {
343     if (p->string)
344         free(p->string);
345     free(p);
346 }

349 void
350 man_node_delete(struct man *m, struct man_node *p)
351 {
353     while (p->child)
354         man_node_delete(m, p->child);

356     man_node_unlink(m, p);
357     man_node_free(p);
358 }

360 int
361 man_addeqn(struct man *m, const struct eqn *ep)
362 {
363     struct man_node *n;

365     assert( ! (MAN_HALT & m->flags));

367     n = man_node_alloc(m, ep->ln, ep->pos, MAN_EQN, MAN_MAX);
368     n->eqn = ep;

370     if ( ! man_node_append(m, n))
371         return(0);

373     m->next = MAN_NEXT_SIBLING;
374     return(man_descope(m, ep->ln, ep->pos));
375 }

377 int
378 man_addspan(struct man *m, const struct tbl_span *sp)
379 {
380     struct man_node *n;

382     assert( ! (MAN_HALT & m->flags));

384     n = man_node_alloc(m, sp->line, 0, MAN_TBL, MAN_MAX);
385     n->span = sp;

```

```

387     if ( ! man_node_append(m, n))
388         return(0);

390     m->next = MAN_NEXT_SIBLING;
391     return(man_descope(m, sp->line, 0));
392 }

394 static int
395 man_descope(struct man *m, int line, int offs)
396 {
397     /*
398     * Co-ordinate what happens with having a next-line scope open:
399     * first close out the element scope (if applicable), then close
400     * out the block scope (also if applicable).
401     */

403     if (MAN_ELINE & m->flags) {
404         m->flags &= ~MAN_ELINE;
405         if ( ! man_unscope(m, m->last->parent, MANDOCERR_MAX))
406             return(0);
407     }

409     if ( ! (MAN_BLINE & m->flags))
410         return(1);
411     m->flags &= ~MAN_BLINE;

413     if ( ! man_unscope(m, m->last->parent, MANDOCERR_MAX))
414         return(0);
415     return(man_body_alloc(m, line, offs, m->last->tok));
416 }

418 static int
419 man_ptext(struct man *m, int line, char *buf, int offs)
420 {
421     int         i;

423     /* Literal free-form text whitespace is preserved. */

425     if (MAN_LITERAL & m->flags) {
426         if ( ! man_word_alloc(m, line, offs, buf + offs))
427             return(0);
428         return(man_descope(m, line, offs));
429     }

431     /* Pump blank lines directly into the backend. */

433     for (i = offs; ' ' == buf[i]; i++)
434         /* Skip leading whitespace. */ ;

436     if ('\0' == buf[i]) {
437         /* Allocate a blank entry. */
438         if ( ! man_word_alloc(m, line, offs, ""))
439             return(0);
440         return(man_descope(m, line, offs));
441     }

443     /*
444     * Warn if the last un-escaped character is whitespace. Then
445     * strip away the remaining spaces (tabs stay!).
446     */

448     i = (int)strlen(buf);
449     assert(i);

451     if (' ' == buf[i - 1] || '\t' == buf[i - 1]) {
452         if (i > 1 && '\\\0' != buf[i - 2])

```

```

453         man_pmsg(m, line, i - 1, MANDOCERR_EOLNSPACE);

455         for (--i; i && ' ' == buf[i]; i--)
456             /* Spin back to non-space. */ ;

458         /* Jump ahead of escaped whitespace. */
459         i += '\\\0' == buf[i] ? 2 : 1;

461         buf[i] = '\0';
462     }

464     if ( ! man_word_alloc(m, line, offs, buf + offs))
465         return(0);

467     /*
468     * End-of-sentence check. If the last character is an unescaped
469     * EOS character, then flag the node as being the end of a
470     * sentence. The front-end will know how to interpret this.
471     */

473     assert(i);
474     if (mandoc_eos(buf, (size_t)i, 0))
475         m->last->flags |= MAN_EOS;

477     return(man_descope(m, line, offs));
478 }

480 static int
481 man_pmacro(struct man *m, int ln, char *buf, int offs)
482 {
483     int         i, ppos;
484     enum mant   tok;
485     char        mac[5];
486     struct man_node *n;

488     if ("'" == buf[offs]) {
489         man_pmsg(m, ln, offs, MANDOCERR_BADCOMMENT);
490         return(1);
491     } else if ('\0' == buf[offs])
492         return(1);

494     ppos = offs;

496     /*
497     * Copy the first word into a nil-terminated buffer.
498     * Stop copying when a tab, space, or eoln is encountered.
499     */

501     i = 0;
502     while (i < 4 && '\0' != buf[offs] &&
503           ' ' != buf[offs] && '\t' != buf[offs])
504         mac[i++] = buf[offs++];

506     mac[i] = '\0';

508     tok = (i > 0 && i < 4) ? man_hash_find(mac) : MAN_MAX;

510     if (MAN_MAX == tok) {
511         mandoc_vmsg(MANDOCERR_MACRO, m->parse, ln,
512                   ppos, "%s", buf + ppos - 1);
513         return(1);
514     }

516     /* The macro is sane. Jump to the next word. */

518     while (buf[offs] && ' ' == buf[offs])

```

```

519         offs++;
521     /*
522     * Trailing whitespace. Note that tabs are allowed to be passed
523     * into the parser as "text", so we only warn about spaces here.
524     */
526     if ('\0' == buf[offs] && ' ' == buf[offs - 1])
527         man_pmsg(m, ln, offs - 1, MANDOCERR_EOLNSPACE);
529     /*
530     * Remove prior ELINE macro, as it's being clobbered by a new
531     * macro. Note that NSCOPEd macros do not close out ELINE
532     * macros---they don't print text---so we let those slip by.
533     */
535     if ( ! (MAN_NSCOPEd & man_macros[tok].flags) &&
536         m->flags & MAN_ELINE) {
537         n = m->last;
538         assert(MAN_TEXT != n->type);
540         /* Remove repeated NSCOPEd macros causing ELINE. */
542         if (MAN_NSCOPEd & man_macros[n->tok].flags)
543             n = n->parent;
545         mandoc_vmsg(MANDOCERR_LINESCOPE, m->parse, n->line,
546                   n->pos, "%s breaks %s", man_macronames[tok],
547                   man_macronames[n->tok]);
549         man_node_delete(m, n);
550         m->flags &= ~MAN_ELINE;
551     }
553     /*
554     * Remove prior BLINE macro that is being clobbered.
555     */
556     if ((m->flags & MAN_BLINE) &&
557         (MAN_BSCOPE & man_macros[tok].flags)) {
558         n = m->last;
560         /* Might be a text node like 8 in
561         * .TP 8
562         * .SH foo
563         */
564         if (MAN_TEXT == n->type)
565             n = n->parent;
567         /* Remove element that didn't end BLINE, if any. */
568         if ( ! (MAN_BSCOPE & man_macros[n->tok].flags))
569             n = n->parent;
571         assert(MAN_HEAD == n->type);
572         n = n->parent;
573         assert(MAN_BLOCK == n->type);
574         assert(MAN_SCOPEd & man_macros[n->tok].flags);
576         mandoc_vmsg(MANDOCERR_LINESCOPE, m->parse, n->line,
577                   n->pos, "%s breaks %s", man_macronames[tok],
578                   man_macronames[n->tok]);
580         man_node_delete(m, n);
581         m->flags &= ~MAN_BLINE;
582     }
584     /*

```

```

585     * Save the fact that we're in the next-line for a block. In
586     * this way, embedded roff instructions can "remember" state
587     * when they exit.
588     */
590     if (MAN_BLINE & m->flags)
591         m->flags |= MAN_BPLINE;
593     /* Call to handler... */
595     assert(man_macros[tok].fp);
596     if ( ! (*man_macros[tok].fp)(m, tok, ln, ppos, &offs, buf))
597         goto err;
599     /*
600     * We weren't in a block-line scope when entering the
601     * above-parsed macro, so return.
602     */
604     if ( ! (MAN_BPLINE & m->flags)) {
605         m->flags &= ~MAN_ILINE;
606         return(1);
607     }
608     m->flags &= ~MAN_BPLINE;
610     /*
611     * If we're in a block scope, then allow this macro to slip by
612     * without closing scope around it.
613     */
615     if (MAN_ILINE & m->flags) {
616         m->flags &= ~MAN_ILINE;
617         return(1);
618     }
620     /*
621     * If we've opened a new next-line element scope, then return
622     * now, as the next line will close out the block scope.
623     */
625     if (MAN_ELINE & m->flags)
626         return(1);
628     /* Close out the block scope opened in the prior line. */
630     assert(MAN_BLINE & m->flags);
631     m->flags &= ~MAN_BLINE;
633     if ( ! man_unscope(m, m->last->parent, MANDOCERR_MAX))
634         return(0);
635     return(man_body_alloc(m, ln, ppos, m->last->tok));
637 err:    /* Error out. */
639     m->flags |= MAN_HALT;
640     return(0);
641 }
643 /*
644 * Unlink a node from its context. If "m" is provided, the last parse
645 * point will also be adjusted accordingly.
646 */
647 static void
648 man_node_unlink(struct man *m, struct man_node *n)
649 {

```

```
651     /* Adjust siblings. */
652
653     if (n->prev)
654         n->prev->next = n->next;
655     if (n->next)
656         n->next->prev = n->prev;
657
658     /* Adjust parent. */
659
660     if (n->parent) {
661         n->parent->nchild--;
662         if (n->parent->child == n)
663             n->parent->child = n->prev ? n->prev : n->next;
664     }
665
666     /* Adjust parse point, if applicable. */
667
668     if (m && m->last == n) {
669         /*XXX: this can occur when bailing from validation. */
670         /*assert(NULL == n->next);*/
671         if (n->prev) {
672             m->last = n->prev;
673             m->next = MAN_NEXT_SIBLING;
674         } else {
675             m->last = n->parent;
676             m->next = MAN_NEXT_CHILD;
677         }
678     }
679
680     if (m && m->first == n)
681         m->first = NULL;
682 }
683
684 const struct mparse *
685 man_mparse(const struct man *m)
686 {
687
688     assert(m && m->parse);
689     return(m->parse);
690 }
```



```
*****
```

```
2695 Sat Jul 19 19:24:30 2014
```

```
new/usr/src/cmd/mandoc/man.h
```

```
5025 import and use mandoc
```

```
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
```

```
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
```

```
Reviewed by: Robert Mustacchi <rm@joyent.com>
```

```
Reviewed by: Albert Lee <trisk@nexenta.com>
```

```
Approved by: TBD
```

```
*****
```

```
1 /* $Id: man.h,v 1.60 2012/01/03 15:16:24 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef MAN_H
18 #define MAN_H
19
20 enum mant {
21     MAN_br = 0,
22     MAN_TH,
23     MAN_SH,
24     MAN_SS,
25     MAN_TP,
26     MAN_LP,
27     MAN_PP,
28     MAN_P,
29     MAN_IP,
30     MAN_HP,
31     MAN_SM,
32     MAN_SB,
33     MAN_BI,
34     MAN_IB,
35     MAN_BR,
36     MAN_RB,
37     MAN_R,
38     MAN_B,
39     MAN_I,
40     MAN_IR,
41     MAN_RI,
42     MAN_na,
43     MAN_sp,
44     MAN_nI,
45     MAN_fi,
46     MAN_RE,
47     MAN_RS,
48     MAN_DT,
49     MAN_UC,
50     MAN_PD,
51     MAN_AT,
52     MAN_in,
53     MAN_ft,
54     MAN_OP,
55     MAN_MAX
56 };
```

```
58 enum man_type {
59     MAN_TEXT,
60     MAN_ELEM,
61     MAN_ROOT,
62     MAN_BLOCK,
63     MAN_HEAD,
64     MAN_BODY,
65     MAN_TAIL,
66     MAN_TBL,
67     MAN_EQN
68 };
69
70 struct man_meta {
71     char *msec; /* 'TH' section (1, 3p, etc.) */
72     char *date; /* 'TH' normalised date */
73     char *vol; /* 'TH' volume */
74     char *title; /* 'TH' title (e.g., FOO) */
75     char *source; /* 'TH' source (e.g., GNU) */
76 };
77
78 struct man_node {
79     struct man_node *parent; /* parent AST node */
80     struct man_node *child; /* first child AST node */
81     struct man_node *next; /* sibling AST node */
82     struct man_node *prev; /* prior sibling AST node */
83     int nchild; /* number children */
84     int line;
85     int pos;
86     enum mant tok; /* tok or MAN_MAX if none */
87     int flags;
88     #define MAN_VALID (1 << 0) /* has been validated */
89     #define MAN_EOS (1 << 2) /* at sentence boundary */
90     #define MAN_LINE (1 << 3) /* first macro/text on line */
91     enum man_type type; /* AST node type */
92     char *string; /* TEXT node argument */
93     struct man_node *head; /* BLOCK node HEAD ptr */
94     struct man_node *tail; /* BLOCK node TAIL ptr */
95     struct man_node *body; /* BLOCK node BODY ptr */
96     const struct tbl_span *span; /* TBL */
97     const struct eqn *eqn; /* EQN */
98 };
99
100 /* Names of macros. Index is enum mant. */
101 extern const char *const *man_macronames;
102
103 __BEGIN_DECLS
104
105 struct man;
106
107 const struct man_node *man_node(const struct man *);
108 const struct man_meta *man_meta(const struct man *);
109 const struct mparse *man_mparse(const struct man *);
110
111 __END_DECLS
112
113 #endif /* !MAN_H */
```

```

*****
2586 Sat Jul 19 19:24:30 2014
new/usr/src/cmd/mandoc/man_hash.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: man_hash.c,v 1.25 2011/07/24 18:15:14 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
21 #include <sys/types.h>
23 #include <assert.h>
24 #include <ctype.h>
25 #include <limits.h>
26 #include <stdlib.h>
27 #include <string.h>
29 #include "man.h"
30 #include "mandoc.h"
31 #include "libman.h"
33 #define HASH_DEPTH      6
35 #define HASH_ROW(x) do { \
36     if (isupper((unsigned char)(x))) \
37         (x) -= 65; \
38     else \
39         (x) -= 97; \
40     (x) *= HASH_DEPTH; \
41 } while (/* CONSTCOND */ 0)
43 /*
44  * Lookup table is indexed first by lower-case first letter (plus one
45  * for the period, which is stored in the last row), then by lower or
46  * uppercase second letter. Buckets correspond to the index of the
47  * macro (the integer value of the enum stored as a char to save a bit
48  * of space).
49  */
50 static unsigned char    table[26 * HASH_DEPTH];
52 /*
53  * XXX - this hash has global scope, so if intended for use as a library
54  * with multiple callers, it will need re-invocation protection.
55  */
56 void

```

```

57 man_hash_init(void)
58 {
59     int            i, j, x;
61     memset(table, UCHAR_MAX, sizeof(table));
63     assert(/* LINTED */
64            MAN_MAX < UCHAR_MAX);
66     for (i = 0; i < (int)MAN_MAX; i++) {
67         x = man_macronames[i][0];
69         assert(isalpha((unsigned char)x));
71         HASH_ROW(x);
73         for (j = 0; j < HASH_DEPTH; j++)
74             if (UCHAR_MAX == table[x + j]) {
75                 table[x + j] = (unsigned char)i;
76                 break;
77             }
79         assert(j < HASH_DEPTH);
80     }
81 }
84 enum mant
85 man_hash_find(const char *tmp)
86 {
87     int            x, y, i;
88     enum mant      tok;
90     if ('\0' == (x = tmp[0]))
91         return(MAN_MAX);
92     if (! (isalpha((unsigned char)x)))
93         return(MAN_MAX);
95     HASH_ROW(x);
97     for (i = 0; i < HASH_DEPTH; i++) {
98         if (UCHAR_MAX == (y = table[x + i]))
99             return(MAN_MAX);
101         tok = (enum mant)y;
102         if (0 == strcmp(tmp, man_macronames[tok]))
103             return(tok);
104     }
106     return(MAN_MAX);
107 }

```

```
*****
```

```
13891 Sat Jul 19 19:24:30 2014
```

```
new/usr/src/cmd/mandoc/man_html.c
```

```
5025 import and use mandoc
```

```
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
```

```
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
```

```
Reviewed by: Robert Mustacchi <rm@joyent.com>
```

```
Reviewed by: Albert Lee <trisk@nexenta.com>
```

```
Approved by: TBD
```

```
*****
```

```
1 /* $Id: man_html.c,v 1.86 2012/01/03 15:16:24 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <sys/types.h>
22
23 #include <assert.h>
24 #include <ctype.h>
25 #include <stdio.h>
26 #include <stdlib.h>
27 #include <string.h>
28
29 #include "mandoc.h"
30 #include "out.h"
31 #include "html.h"
32 #include "man.h"
33 #include "main.h"
34
35 /* TODO: preserve ident widths. */
36 /* FIXME: have PD set the default vspace width. */
37
38 #define INDENT 5
39
40 #define MAN_ARGS const struct man_meta *m, \
41                 const struct man_node *n, \
42                 struct mhtml *mh, \
43                 struct html *h
44
45 struct mhtml {
46     int fl;
47 #define MANH_LITERAL (1 << 0) /* literal context */
48 };
49
50 struct htmlman {
51     int (*pre)(MAN_ARGS);
52     int (*post)(MAN_ARGS);
53 };
54
55 static void print_bvspace(struct html *,
56                          const struct man_node *);
```

```
57 static void print_man(MAN_ARGS);
58 static void print_man_head(MAN_ARGS);
59 static void print_man_nodelist(MAN_ARGS);
60 static void print_man_node(MAN_ARGS);
61 static int a2width(const struct man_node *,
62                   struct roffsu *);
63 static int man_B_pre(MAN_ARGS);
64 static int man_HP_pre(MAN_ARGS);
65 static int man_IP_pre(MAN_ARGS);
66 static int man_I_pre(MAN_ARGS);
67 static int man_OP_pre(MAN_ARGS);
68 static int man_PP_pre(MAN_ARGS);
69 static int man_RS_pre(MAN_ARGS);
70 static int man_SH_pre(MAN_ARGS);
71 static int man_SM_pre(MAN_ARGS);
72 static int man_SS_pre(MAN_ARGS);
73 static int man_alt_pre(MAN_ARGS);
74 static int man_br_pre(MAN_ARGS);
75 static int man_ign_pre(MAN_ARGS);
76 static int man_in_pre(MAN_ARGS);
77 static int man_literal_pre(MAN_ARGS);
78 static void man_root_post(MAN_ARGS);
79 static void man_root_pre(MAN_ARGS);
80
81 static const struct htmlman mans[MAN_MAX] = {
82     { man_br_pre, NULL }, /* br */
83     { NULL, NULL }, /* TH */
84     { man_SH_pre, NULL }, /* SH */
85     { man_SS_pre, NULL }, /* SS */
86     { man_IP_pre, NULL }, /* TP */
87     { man_PP_pre, NULL }, /* LP */
88     { man_PP_pre, NULL }, /* PP */
89     { man_PP_pre, NULL }, /* P */
90     { man_IP_pre, NULL }, /* IP */
91     { man_HP_pre, NULL }, /* HP */
92     { man_SM_pre, NULL }, /* SM */
93     { man_SM_pre, NULL }, /* SB */
94     { man_alt_pre, NULL }, /* BI */
95     { man_alt_pre, NULL }, /* IB */
96     { man_alt_pre, NULL }, /* BR */
97     { man_alt_pre, NULL }, /* RB */
98     { NULL, NULL }, /* R */
99     { man_B_pre, NULL }, /* B */
100    { man_I_pre, NULL }, /* I */
101    { man_alt_pre, NULL }, /* IR */
102    { man_alt_pre, NULL }, /* RI */
103    { man_ign_pre, NULL }, /* na */
104    { man_br_pre, NULL }, /* sp */
105    { man_literal_pre, NULL }, /* nf */
106    { man_literal_pre, NULL }, /* fi */
107    { NULL, NULL }, /* RE */
108    { man_RS_pre, NULL }, /* RS */
109    { man_ign_pre, NULL }, /* DT */
110    { man_ign_pre, NULL }, /* UC */
111    { man_ign_pre, NULL }, /* PD */
112    { man_ign_pre, NULL }, /* AT */
113    { man_in_pre, NULL }, /* in */
114    { man_ign_pre, NULL }, /* ft */
115    { man_OP_pre, NULL }, /* OP */
116 };
117
118 /*
119  * Printing leading vertical space before a block.
120  * This is used for the paragraph macros.
121  * The rules are pretty simple, since there's very little nesting going
122  * on here. Basically, if we're the first within another block (SS/SH),
```

```

123 * then don't emit vertical space.  If we are (RS), then do.  If not the
124 * first, print it.
125 */
126 static void
127 print_bvspace(struct html *h, const struct man_node *n)
128 {
129
130     if (n->body && n->body->child)
131         if (MAN_TBL == n->body->child->type)
132             return;
133
134     if (MAN_ROOT == n->parent->type || MAN_RS != n->parent->tok)
135         if (NULL == n->prev)
136             return;
137
138     print_otag(h, TAG_P, 0, NULL);
139 }
140
141 void
142 html_man(void *arg, const struct man *m)
143 {
144     struct mhtml     mh;
145
146     memset(&mh, 0, sizeof(struct mhtml));
147     print_man(man_meta(m), man_node(m), &mh, (struct html *)arg);
148     putchar('\n');
149 }
150
151 static void
152 print_man(MAN_ARGS)
153 {
154     struct tag      *t, *tt;
155     struct htmlpair tag;
156
157     PAIR_CLASS_INIT(&tag, "mandoc");
158
159     if (! (HTML_FRAGMENT & h->oflags)) {
160         print_gen_decls(h);
161         t = print_otag(h, TAG_HTML, 0, NULL);
162         tt = print_otag(h, TAG_HEAD, 0, NULL);
163         print_man_head(m, n, mh, h);
164         print_tagq(h, tt);
165         print_otag(h, TAG_BODY, 0, NULL);
166         print_otag(h, TAG_DIV, 1, &tag);
167     } else
168         t = print_otag(h, TAG_DIV, 1, &tag);
169
170     print_man_nodelist(m, n, mh, h);
171     print_tagq(h, t);
172 }
173
174 /* ARGSUSED */
175 static void
176 print_man_head(MAN_ARGS)
177 {
178
179     print_gen_head(h);
180     assert(m->title);
181     assert(m->msec);
182     bufcat_fmt(h, "%s(%s)", m->title, m->msec);
183     print_otag(h, TAG_TITLE, 0, NULL);
184     print_text(h, h->buf);
185 }
186

```

```

189 static void
190 print_man_nodelist(MAN_ARGS)
191 {
192
193     print_man_node(m, n, mh, h);
194     if (n->next)
195         print_man_nodelist(m, n->next, mh, h);
196 }
197
198
199 static void
200 print_man_node(MAN_ARGS)
201 {
202     int             child;
203     struct tag      *t;
204
205     child = 1;
206     t = h->tags.head;
207
208     switch (n->type) {
209     case (MAN_ROOT):
210         man_root_pre(m, n, mh, h);
211         break;
212     case (MAN_TEXT):
213         /*
214          * If we have a blank line, output a vertical space.
215          * If we have a space as the first character, break
216          * before printing the line's data.
217          */
218         if ('\0' == *n->string) {
219             print_otag(h, TAG_P, 0, NULL);
220             return;
221         }
222
223         if (' ' == *n->string && MAN_LINE & n->flags)
224             print_otag(h, TAG_BR, 0, NULL);
225         else if (MANH_LITERAL & mh->fl && n->prev)
226             print_otag(h, TAG_BR, 0, NULL);
227
228         print_text(h, n->string);
229         return;
230     case (MAN_EQN):
231         print_eqn(h, n->eqn);
232         break;
233     case (MAN_TBL):
234         /*
235          * This will take care of initialising all of the table
236          * state data for the first table, then tearing it down
237          * for the last one.
238          */
239         print_tbl(h, n->span);
240         return;
241     default:
242         /*
243          * Close out scope of font prior to opening a macro
244          * scope.
245          */
246         if (HTMLFONT_NONE != h->metac) {
247             h->metal = h->metac;
248             h->metac = HTMLFONT_NONE;
249         }
250
251         /*
252          * Close out the current table, if it's open, and unset
253          * the "meta" table state.  This will be reopened on the
254          * next table element.

```

```

255     */
256     if (h->tblt) {
257         print_tblclose(h);
258         t = h->tags.head;
259     }
260     if (mans[n->tok].pre)
261         child = (*mans[n->tok].pre)(m, n, mh, h);
262     break;
263 }

265 if (child && n->child)
266     print_man_nodelist(m, n->child, mh, h);

268 /* This will automatically close out any font scope. */
269 print_stagq(h, t);

271 switch (n->type) {
272 case (MAN_ROOT):
273     man_root_post(m, n, mh, h);
274     break;
275 case (MAN_EQN):
276     break;
277 default:
278     if (mans[n->tok].post)
279         (*mans[n->tok].post)(m, n, mh, h);
280     break;
281 }
282 }

285 static int
286 a2width(const struct man_node *n, struct roffsu *su)
287 {

289     if (MAN_TEXT != n->type)
290         return(0);
291     if (a2roffsu(n->string, su, SCALE_BU))
292         return(1);

294     return(0);
295 }

298 /* ARGSUSED */
299 static void
300 man_root_pre(MAN_ARGS)
301 {
302     struct htmlpair tag[3];
303     struct tag *t, *tt;
304     char b[BUFSIZ], title[BUFSIZ];

306     b[0] = 0;
307     if (m->vol)
308         (void)strlcat(b, m->vol, BUFSIZ);

310     assert(m->title);
311     assert(m->msec);
312     snprintf(title, BUFSIZ - 1, "%s(%s)", m->title, m->msec);

314     PAIR_SUMMARY_INIT(&tag[0], "Document Header");
315     PAIR_CLASS_INIT(&tag[1], "head");
316     PAIR_INIT(&tag[2], ATTR_WIDTH, "100%");
317     t = print_otag(h, TAG_TABLE, 3, tag);
318     PAIR_INIT(&tag[0], ATTR_WIDTH, "30%");
319     print_otag(h, TAG_COL, 1, tag);
320     print_otag(h, TAG_COL, 1, tag);

```

```

321     print_otag(h, TAG_COL, 1, tag);

323     print_otag(h, TAG_TBODY, 0, NULL);

325     tt = print_otag(h, TAG_TR, 0, NULL);

327     PAIR_CLASS_INIT(&tag[0], "head-ltitle");
328     print_otag(h, TAG_TD, 1, tag);
329     print_text(h, title);
330     print_stagq(h, tt);

332     PAIR_CLASS_INIT(&tag[0], "head-vol");
333     PAIR_INIT(&tag[1], ATTR_ALIGN, "center");
334     print_otag(h, TAG_TD, 2, tag);
335     print_text(h, b);
336     print_stagq(h, tt);

338     PAIR_CLASS_INIT(&tag[0], "head-rttitle");
339     PAIR_INIT(&tag[1], ATTR_ALIGN, "right");
340     print_otag(h, TAG_TD, 2, tag);
341     print_text(h, title);
342     print_tagq(h, t);
343 }

346 /* ARGSUSED */
347 static void
348 man_root_post(MAN_ARGS)
349 {
350     struct htmlpair tag[3];
351     struct tag *t, *tt;

353     PAIR_SUMMARY_INIT(&tag[0], "Document Footer");
354     PAIR_CLASS_INIT(&tag[1], "foot");
355     PAIR_INIT(&tag[2], ATTR_WIDTH, "100%");
356     t = print_otag(h, TAG_TABLE, 3, tag);
357     PAIR_INIT(&tag[0], ATTR_WIDTH, "50%");
358     print_otag(h, TAG_COL, 1, tag);
359     print_otag(h, TAG_COL, 1, tag);

361     tt = print_otag(h, TAG_TR, 0, NULL);

363     PAIR_CLASS_INIT(&tag[0], "foot-date");
364     print_otag(h, TAG_TD, 1, tag);

366     assert(m->date);
367     print_text(h, m->date);
368     print_stagq(h, tt);

370     PAIR_CLASS_INIT(&tag[0], "foot-os");
371     PAIR_INIT(&tag[1], ATTR_ALIGN, "right");
372     print_otag(h, TAG_TD, 2, tag);

374     if (m->source)
375         print_text(h, m->source);
376     print_tagq(h, t);
377 }

380 /* ARGSUSED */
381 static int
382 man_br_pre(MAN_ARGS)
383 {
384     struct roffsu su;
385     struct htmlpair tag;

```

```

387     SCALE_VS_INIT(&su, 1);
389     if (MAN_sp == n->tok) {
390         if (NULL != (n = n->child))
391             if (! a2roffsu(n->string, &su, SCALE_VS))
392                 SCALE_VS_INIT(&su, atoi(n->string));
393     } else
394         su.scale = 0;
396     bufinit(h);
397     bufcat_su(h, "height", &su);
398     PAIR_STYLE_INIT(&tag, h);
399     print_otag(h, TAG_DIV, 1, &tag);
401     /* So the div isn't empty: */
402     print_text(h, "\\~");
404     return(0);
405 }
407 /* ARGSUSED */
408 static int
409 man_SH_pre(MAN_ARGS)
410 {
411     struct htmlpair tag;
413     if (MAN_BLOCK == n->type) {
414         mh->fl &= ~MANH_LITERAL;
415         PAIR_CLASS_INIT(&tag, "section");
416         print_otag(h, TAG_DIV, 1, &tag);
417         return(1);
418     } else if (MAN_BODY == n->type)
419         return(1);
421     print_otag(h, TAG_H1, 0, NULL);
422     return(1);
423 }
425 /* ARGSUSED */
426 static int
427 man_alt_pre(MAN_ARGS)
428 {
429     const struct man_node *nn;
430     int i, savelit;
431     enum htmltag fp;
432     struct tag *t;
434     if ((savelit = mh->fl & MANH_LITERAL))
435         print_otag(h, TAG_BR, 0, NULL);
437     mh->fl &= ~MANH_LITERAL;
439     for (i = 0, nn = n->child; nn; nn = nn->next, i++) {
440         t = NULL;
441         switch (n->tok) {
442             case (MAN_BI):
443                 fp = i % 2 ? TAG_I : TAG_B;
444                 break;
445             case (MAN_IB):
446                 fp = i % 2 ? TAG_B : TAG_I;
447                 break;
448             case (MAN_RI):
449                 fp = i % 2 ? TAG_I : TAG_MAX;
450                 break;
451             case (MAN_IR):
452                 fp = i % 2 ? TAG_MAX : TAG_I;

```

```

453         break;
454     case (MAN_BR):
455         fp = i % 2 ? TAG_MAX : TAG_B;
456         break;
457     case (MAN_RB):
458         fp = i % 2 ? TAG_B : TAG_MAX;
459         break;
460     default:
461         abort();
462         /* NOTREACHED */
463     }
465     if (i)
466         h->flags |= HTML_NOSPACE;
468     if (TAG_MAX != fp)
469         t = print_otag(h, fp, 0, NULL);
471     print_man_node(m, nn, mh, h);
473     if (t)
474         print_tagq(h, t);
475 }
477     if (savelit)
478         mh->fl |= MANH_LITERAL;
480     return(0);
481 }
483 /* ARGSUSED */
484 static int
485 man_SM_pre(MAN_ARGS)
486 {
487     print_otag(h, TAG_SMALL, 0, NULL);
488     if (MAN_SB == n->tok)
489         print_otag(h, TAG_B, 0, NULL);
490     return(1);
491 }
492 }
494 /* ARGSUSED */
495 static int
496 man_SS_pre(MAN_ARGS)
497 {
498     struct htmlpair tag;
500     if (MAN_BLOCK == n->type) {
501         mh->fl &= ~MANH_LITERAL;
502         PAIR_CLASS_INIT(&tag, "subsection");
503         print_otag(h, TAG_DIV, 1, &tag);
504         return(1);
505     } else if (MAN_BODY == n->type)
506         return(1);
508     print_otag(h, TAG_H2, 0, NULL);
509     return(1);
510 }
512 /* ARGSUSED */
513 static int
514 man_PP_pre(MAN_ARGS)
515 {
517     if (MAN_HEAD == n->type)
518         return(0);

```

```

519     else if (MAN_BLOCK == n->type)
520         print_bvspace(h, n);

522     return(1);
523 }

525 /* ARGSUSED */
526 static int
527 man_IP_pre(MAN_ARGS)
528 {
529     const struct man_node *nn;

531     if (MAN_BODY == n->type) {
532         print_otag(h, TAG_DD, 0, NULL);
533         return(1);
534     } else if (MAN_HEAD != n->type) {
535         print_otag(h, TAG_DL, 0, NULL);
536         return(1);
537     }

539     /* FIXME: width specification. */

541     print_otag(h, TAG_DT, 0, NULL);

543     /* For IP, only print the first header element. */

545     if (MAN_IP == n->tok && n->child)
546         print_man_node(m, n->child, mh, h);

548     /* For TP, only print next-line header elements. */

550     if (MAN_TP == n->tok)
551         for (nn = n->child; nn; nn = nn->next)
552             if (nn->line > n->line)
553                 print_man_node(m, nn, mh, h);

555     return(0);
556 }

558 /* ARGSUSED */
559 static int
560 man_HP_pre(MAN_ARGS)
561 {
562     struct htmlpair tag;
563     struct roffsu su;
564     const struct man_node *np;

566     if (MAN_HEAD == n->type)
567         return(0);
568     else if (MAN_BLOCK != n->type)
569         return(1);

571     np = n->head->child;

573     if (NULL == np || ! a2width(np, &su))
574         SCALE_HS_INIT(&su, INDENT);

576     bufinit(h);

578     print_bvspace(h, n);
579     bufcat_su(h, "margin-left", &su);
580     su.scale = -su.scale;
581     bufcat_su(h, "text-indent", &su);
582     PAIR_STYLE_INIT(&tag, h);
583     print_otag(h, TAG_P, 1, &tag);
584     return(1);

```

```

585 }

587 /* ARGSUSED */
588 static int
589 man_OP_pre(MAN_ARGS)
590 {
591     struct tag *tt;
592     struct htmlpair tag;

594     print_text(h, "[");
595     h->flags |= HTML_NOSPACE;
596     PAIR_CLASS_INIT(&tag, "opt");
597     tt = print_otag(h, TAG_SPAN, 1, &tag);

599     if (NULL != (n = n->child)) {
600         print_otag(h, TAG_B, 0, NULL);
601         print_text(h, n->string);
602     }

604     print_stagq(h, tt);

606     if (NULL != n && NULL != n->next) {
607         print_otag(h, TAG_I, 0, NULL);
608         print_text(h, n->next->string);
609     }

611     print_stagq(h, tt);
612     h->flags |= HTML_NOSPACE;
613     print_text(h, "]");
614     return(0);
615 }

618 /* ARGSUSED */
619 static int
620 man_B_pre(MAN_ARGS)
621 {
623     print_otag(h, TAG_B, 0, NULL);
624     return(1);
625 }

627 /* ARGSUSED */
628 static int
629 man_I_pre(MAN_ARGS)
630 {
631     print_otag(h, TAG_I, 0, NULL);
632     return(1);
633 }
634 }

636 /* ARGSUSED */
637 static int
638 man_literal_pre(MAN_ARGS)
639 {
641     if (MAN_nf != n->tok) {
642         print_otag(h, TAG_BR, 0, NULL);
643         mh->fl &= ~MANH_LITERAL;
644     } else
645         mh->fl |= MANH_LITERAL;

647     return(0);
648 }

650 /* ARGSUSED */

```

```
651 static int
652 man_in_pre(MAN_ARGS)
653 {
654
655     print_otag(h, TAG_BR, 0, NULL);
656     return(0);
657 }
658
659 /* ARGSUSED */
660 static int
661 man_ign_pre(MAN_ARGS)
662 {
663
664     return(0);
665 }
666
667 /* ARGSUSED */
668 static int
669 man_RS_pre(MAN_ARGS)
670 {
671     struct htmlpair tag;
672     struct roffsu su;
673
674     if (MAN_HEAD == n->type)
675         return(0);
676     else if (MAN_BODY == n->type)
677         return(1);
678
679     SCALE_HS_INIT(&su, INDENT);
680     if (n->head->child)
681         a2width(n->head->child, &su);
682
683     bufinit(h);
684     bufcat_su(h, "margin-left", &su);
685     PAIR_STYLE_INIT(&tag, h);
686     print_otag(h, TAG_DIV, 1, &tag);
687     return(1);
688 }
```



```

*****
10870 Sat Jul 19 19:24:30 2014
new/usr/src/cmd/mandoc/man_macro.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: man_macro.c,v 1.71 2012/01/03 15:16:24 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <assert.h>
22 #include <ctype.h>
23 #include <stdlib.h>
24 #include <string.h>
25
26 #include "man.h"
27 #include "mandoc.h"
28 #include "libmandoc.h"
29 #include "libman.h"
30
31 enum      rew {
32     REW_REWIND,
33     REW_NOHALT,
34     REW_HALT
35 };
36
37 static int      blk_close(MACRO_PROT_ARGS);
38 static int      blk_exp(MACRO_PROT_ARGS);
39 static int      blk_imp(MACRO_PROT_ARGS);
40 static int      in_line_eoln(MACRO_PROT_ARGS);
41 static int      man_args(struct man *, int,
42                          int *, char *, char **);
43
44 static int      rew_scope(enum man_type,
45                          struct man *, enum mant);
46 static enum rew rew_dohalt(enum mant, enum man_type,
47                          const struct man_node *);
48 static enum rew rew_block(enum mant, enum man_type,
49                          const struct man_node *);
50 static void      rew_warn(struct man *,
51                          struct man_node *, enum mandocerr);
52
53 const struct man_macro __man_macros[MAN_MAX] = {
54     { in_line_eoln, MAN_NSSCOPED }, /* br */
55     { in_line_eoln, MAN_BSCOPE }, /* th */
56     { blk_imp, MAN_BSCOPE | MAN_SCOPED }, /* sh */

```

```

57     { blk_imp, MAN_BSCOPE | MAN_SCOPED }, /* ss */
58     { blk_imp, MAN_BSCOPE | MAN_SCOPED | MAN_FSCOPED }, /* tp */
59     { blk_imp, MAN_BSCOPE }, /* lp */
60     { blk_imp, MAN_BSCOPE }, /* pp */
61     { blk_imp, MAN_BSCOPE }, /* p */
62     { blk_imp, MAN_BSCOPE }, /* ip */
63     { blk_imp, MAN_BSCOPE }, /* hp */
64     { in_line_eoln, MAN_SCOPED }, /* sm */
65     { in_line_eoln, MAN_SCOPED }, /* sb */
66     { in_line_eoln, 0 }, /* bi */
67     { in_line_eoln, 0 }, /* ib */
68     { in_line_eoln, 0 }, /* br */
69     { in_line_eoln, 0 }, /* rb */
70     { in_line_eoln, MAN_SCOPED }, /* r */
71     { in_line_eoln, MAN_SCOPED }, /* b */
72     { in_line_eoln, MAN_SCOPED }, /* i */
73     { in_line_eoln, 0 }, /* ir */
74     { in_line_eoln, 0 }, /* ri */
75     { in_line_eoln, MAN_NSSCOPED }, /* na */
76     { in_line_eoln, MAN_NSSCOPED }, /* sp */
77     { in_line_eoln, MAN_BSCOPE }, /* nf */
78     { in_line_eoln, MAN_BSCOPE }, /* fi */
79     { blk_close, 0 }, /* re */
80     { blk_exp, MAN_EXPLICIT }, /* rs */
81     { in_line_eoln, 0 }, /* dt */
82     { in_line_eoln, 0 }, /* uc */
83     { in_line_eoln, 0 }, /* pd */
84     { in_line_eoln, 0 }, /* at */
85     { in_line_eoln, 0 }, /* in */
86     { in_line_eoln, 0 }, /* ft */
87     { in_line_eoln, 0 }, /* op */
88 };
89
90 const struct man_macro * const man_macros = __man_macros;
91
92
93 /*
94  * Warn when "n" is an explicit non-roff macro.
95  */
96 static void
97 rew_warn(struct man *m, struct man_node *n, enum mandocerr er)
98 {
99
100     if (er == MANDOCERR_MAX || MAN_BLOCK != n->type)
101         return;
102     if (MAN_VALID & n->flags)
103         return;
104     if (! (MAN_EXPLICIT & man_macros[n->tok].flags))
105         return;
106
107     assert(er < MANDOCERR_FATAL);
108     man_nmsg(m, n, er);
109 }
110
111
112 /*
113  * Rewind scope. If a code "er" != MANDOCERR_MAX has been provided, it
114  * will be used if an explicit block scope is being closed out.
115  */
116 int
117 man_unscope(struct man *m, const struct man_node *to,
118             enum mandocerr er)
119 {
120     struct man_node *n;
121
122     assert(to);

```

```

124     m->next = MAN_NEXT_SIBLING;

126     /* LINTED */
127     while (m->last != to) {
128         /*
129          * Save the parent here, because we may delete the
130          * m->last node in the post-validation phase and reset
131          * it to m->last->parent, causing a step in the closing
132          * out to be lost.
133          */
134         n = m->last->parent;
135         rew_warn(m, m->last, er);
136         if ( ! man_valid_post(m))
137             return(0);
138         m->last = n;
139         assert(m->last);
140     }

142     rew_warn(m, m->last, er);
143     if ( ! man_valid_post(m))
144         return(0);

146     return(1);
147 }

150 static enum rew
151 rew_block(enum mant tok, enum man_type type, const struct man_node *n)
152 {

154     if (MAN_BLOCK == type && tok == n->parent->tok &&
155         MAN_BODY == n->parent->type)
156         return(REW_REWIND);
157     return(tok == n->tok ? REW_HALT : REW_NOHALT);
158 }

161 /*
162 * There are three scope levels: scoped to the root (all), scoped to the
163 * section (all less sections), and scoped to subsections (all less
164 * sections and subsections).
165 */
166 static enum rew
167 rew_dohalt(enum mant tok, enum man_type type, const struct man_node *n)
168 {
169     enum rew      c;

171     /* We cannot progress beyond the root ever. */
172     if (MAN_ROOT == n->type)
173         return(REW_HALT);

175     assert(n->parent);

177     /* Normal nodes shouldn't go to the level of the root. */
178     if (MAN_ROOT == n->parent->type)
179         return(REW_REWIND);

181     /* Already-validated nodes should be closed out. */
182     if (MAN_VALID & n->flags)
183         return(REW_NOHALT);

185     /* First: rewind to ourselves. */
186     if (type == n->type && tok == n->tok)
187         return(REW_REWIND);

```

```

189     /*
190     * Next follow the implicit scope-smashings as defined by man.7:
191     * section, sub-section, etc.
192     */

194     switch (tok) {
195     case (MAN_SH):
196         break;
197     case (MAN_SS):
198         /* Rewind to a section, if a block. */
199         if (REW_NOHALT != (c = rew_block(MAN_SH, type, n)))
200             return(c);
201         break;
202     case (MAN_RS):
203         /* Rewind to a subsection, if a block. */
204         if (REW_NOHALT != (c = rew_block(MAN_SS, type, n)))
205             return(c);
206         /* Rewind to a section, if a block. */
207         if (REW_NOHALT != (c = rew_block(MAN_SH, type, n)))
208             return(c);
209         break;
210     default:
211         /* Rewind to an offsetter, if a block. */
212         if (REW_NOHALT != (c = rew_block(MAN_RS, type, n)))
213             return(c);
214         /* Rewind to a subsection, if a block. */
215         if (REW_NOHALT != (c = rew_block(MAN_SS, type, n)))
216             return(c);
217         /* Rewind to a section, if a block. */
218         if (REW_NOHALT != (c = rew_block(MAN_SH, type, n)))
219             return(c);
220         break;
221     }

223     return(REW_NOHALT);
224 }

227 /*
228 * Rewinding entails ascending the parse tree until a coherent point,
229 * for example, the 'SH' macro will close out any intervening 'SS'
230 * scopes. When a scope is closed, it must be validated and actioned.
231 */
232 static int
233 rew_scope(enum man_type type, struct man *m, enum mant tok)
234 {
235     struct man_node *n;
236     enum rew      c;

238     /* LINTED */
239     for (n = m->last; n; n = n->parent) {
240         /*
241          * Whether we should stop immediately (REW_HALT), stop
242          * and rewind until this point (REW_REWIND), or keep
243          * rewinding (REW_NOHALT).
244          */
245         c = rew_dohalt(tok, type, n);
246         if (REW_HALT == c)
247             return(1);
248         if (REW_REWIND == c)
249             break;
250     }

252     /*
253     * Rewind until the current point. Warn if we're a roff
254     * instruction that's mowing over explicit scopes.

```

```

255     */
256     assert(n);

258     return(man_unscope(m, n, MANDOCERR_MAX));
259 }

262 /*
263  * Close out a generic explicit macro.
264  */
265 /* ARGSUSED */
266 int
267 blk_close(MACRO_PROT_ARGS)
268 {
269     enum mant          ntok;
270     const struct man_node *nn;

272     switch (tok) {
273     case (MAN_RE):
274         ntok = MAN_RS;
275         break;
276     default:
277         abort();
278         /* NOTREACHED */
279     }

281     for (nn = m->last->parent; nn; nn = nn->parent)
282         if (ntok == nn->tok)
283             break;

285     if (NULL == nn)
286         man_pmsg(m, line, ppos, MANDOCERR_NOSCOPE);

288     if ( ! rew_scope(MAN_BODY, m, ntok))
289         return(0);
290     if ( ! rew_scope(MAN_BLOCK, m, ntok))
291         return(0);

293     return(1);
294 }

297 /* ARGSUSED */
298 int
299 blk_exp(MACRO_PROT_ARGS)
300 {
301     int          la;
302     char         *p;

304     /*
305      * Close out prior scopes. "Regular" explicit macros cannot be
306      * nested, but we allow roff macros to be placed just about
307      * anywhere.
308      */

310     if ( ! man_block_alloc(m, line, ppos, tok))
311         return(0);
312     if ( ! man_head_alloc(m, line, ppos, tok))
313         return(0);

315     for (;;) {
316         la = *pos;
317         if ( ! man_args(m, line, pos, buf, &p))
318             break;
319         if ( ! man_word_alloc(m, line, la, p))
320             return(0);

```

```

321     }

323     assert(m);
324     assert(tok != MAN_MAX);

326     if ( ! rew_scope(MAN_HEAD, m, tok))
327         return(0);
328     return(man_body_alloc(m, line, ppos, tok));
329 }

333 /*
334  * Parse an implicit-block macro. These contain a MAN_HEAD and a
335  * MAN_BODY contained within a MAN_BLOCK. Rules for closing out other
336  * scopes, such as 'SH' closing out an 'SS', are defined in the rew
337  * routines.
338  */
339 /* ARGSUSED */
340 int
341 blk_imp(MACRO_PROT_ARGS)
342 {
343     int          la;
344     char         *p;
345     struct man_node *n;

347     /* Close out prior scopes. */

349     if ( ! rew_scope(MAN_BODY, m, tok))
350         return(0);
351     if ( ! rew_scope(MAN_BLOCK, m, tok))
352         return(0);

354     /* Allocate new block & head scope. */

356     if ( ! man_block_alloc(m, line, ppos, tok))
357         return(0);
358     if ( ! man_head_alloc(m, line, ppos, tok))
359         return(0);

361     n = m->last;

363     /* Add line arguments. */

365     for (;;) {
366         la = *pos;
367         if ( ! man_args(m, line, pos, buf, &p))
368             break;
369         if ( ! man_word_alloc(m, line, la, p))
370             return(0);
371     }

373     /* Close out head and open body (unless MAN_SCOPE). */

375     if (MAN_SCOPED & man_macros[tok].flags) {
376         /* If we're forcing scope ('TP'), keep it open. */
377         if (MAN_FSCOPED & man_macros[tok].flags) {
378             m->flags |= MAN_BLINE;
379             return(1);
380         } else if (n == m->last) {
381             m->flags |= MAN_BLINE;
382             return(1);
383         }
384     }

386     if ( ! rew_scope(MAN_HEAD, m, tok))

```

```

387         return(0);
388     return(man_body_alloc(m, line, ppos, tok));
389 }

392 /* ARGSUSED */
393 int
394 in_line_eoln(MACRO_PROT_ARGS)
395 {
396     int         la;
397     char        *p;
398     struct man_node *n;

400     if ( ! man_elem_alloc(m, line, ppos, tok))
401         return(0);

403     n = m->last;

405     for (;;) {
406         la = *pos;
407         if ( ! man_args(m, line, pos, buf, &p))
408             break;
409         if ( ! man_word_alloc(m, line, la, p))
410             return(0);
411     }

413     /*
414     * If no arguments are specified and this is MAN_SCOPED (i.e.,
415     * next-line scoped), then set our mode to indicate that we're
416     * waiting for terms to load into our context.
417     */

419     if (n == m->last && MAN_SCOPED & man_macros[tok].flags) {
420         assert( ! (MAN_NSCHOPED & man_macros[tok].flags));
421         m->flags |= MAN_ELINE;
422         return(1);
423     }

425     /* Set ignorable context, if applicable. */

427     if (MAN_NSCHOPED & man_macros[tok].flags) {
428         assert( ! (MAN_SCOPED & man_macros[tok].flags));
429         m->flags |= MAN_ILINE;
430     }

432     assert(MAN_ROOT != m->last->type);
433     m->next = MAN_NEXT_SIBLING;
434
435     /*
436     * Rewind our element scope. Note that when TH is pruned, we'll
437     * be back at the root, so make sure that we don't clobber as
438     * its sibling.
439     */

441     for ( ; m->last; m->last = m->last->parent) {
442         if (m->last == n)
443             break;
444         if (m->last->type == MAN_ROOT)
445             break;
446         if ( ! man_valid_post(m))
447             return(0);
448     }

450     assert(m->last);

452     /*

```

```

453     * Same here regarding whether we're back at the root.
454     */

456     if (m->last->type != MAN_ROOT && ! man_valid_post(m))
457         return(0);

459     return(1);
460 }

463 int
464 man_macroend(struct man *m)
465 {

467     return(man_unscope(m, m->first, MANDOCERR_SCOPEEXIT));
468 }

470 static int
471 man_args(struct man *m, int line, int *pos, char *buf, char **v)
472 {
473     char        *start;

475     assert(*pos);
476     *v = start = buf + *pos;
477     assert(' ' != *start);

479     if ('\0' == *start)
480         return(0);

482     *v = mandoc_getarg(m->parse, v, line, pos);
483     return(1);
484 }

```

```

*****
21486 Sat Jul 19 19:24:30 2014
new/usr/src/cmd/mandoc/man_term.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: man_term.c,v 1.127 2012/01/03 15:16:24 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2010, 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <sys/types.h>
23
24 #include <assert.h>
25 #include <ctype.h>
26 #include <stdio.h>
27 #include <stdlib.h>
28 #include <string.h>
29
30 #include "mandoc.h"
31 #include "out.h"
32 #include "man.h"
33 #include "term.h"
34 #include "main.h"
35
36 #define MAXMARGINS      64 /* maximum number of indented scopes */
37
38 /* FIXME: have PD set the default vspace width. */
39
40 struct mtermp {
41     int          fl;
42 #define MANT_LITERAL    (1 << 0)
43     size_t      lmargin[MAXMARGINS]; /* margins (incl. visible page) */
44     int         lmargincur; /* index of current margin */
45     int         lmarginisz; /* actual number of nested margins */
46     size_t      offset; /* default offset to visible page */
47 };
48
49 #define DECL_ARGS      struct termpp *p, \
50                        struct mtermpp *mt, \
51                        const struct man_node *n, \
52                        const struct man_meta *m
53
54 struct termact {
55     int          (*pre)(DECL_ARGS);
56     void         (*post)(DECL_ARGS);

```

```

57     int          flags;
58 #define MAN_NOTEXT      (1 << 0) /* Never has text children. */
59 };
60
61 static int          a2width(const struct termpp *, const char *);
62 static size_t      a2height(const struct termpp *, const char *);
63
64 static void        print_man_nodelist(DECL_ARGS);
65 static void        print_man_node(DECL_ARGS);
66 static void        print_man_head(struct termpp *, const void *);
67 static void        print_man_foot(struct termpp *, const void *);
68 static void        print_bvspace(struct termpp *,
69                                const struct man_node *);
70
71 static int         pre_B(DECL_ARGS);
72 static int         pre_HP(DECL_ARGS);
73 static int         pre_I(DECL_ARGS);
74 static int         pre_IP(DECL_ARGS);
75 static int         pre_OP(DECL_ARGS);
76 static int         pre_PP(DECL_ARGS);
77 static int         pre_RS(DECL_ARGS);
78 static int         pre_SH(DECL_ARGS);
79 static int         pre_SS(DECL_ARGS);
80 static int         pre_TP(DECL_ARGS);
81 static int         pre_alternate(DECL_ARGS);
82 static int         pre_ft(DECL_ARGS);
83 static int         pre_ign(DECL_ARGS);
84 static int         pre_in(DECL_ARGS);
85 static int         pre_literal(DECL_ARGS);
86 static int         pre_sp(DECL_ARGS);
87
88 static void        post_IP(DECL_ARGS);
89 static void        post_HP(DECL_ARGS);
90 static void        post_RS(DECL_ARGS);
91 static void        post_SH(DECL_ARGS);
92 static void        post_SS(DECL_ARGS);
93 static void        post_TP(DECL_ARGS);
94
95 static const struct termact termacts[MAN_MAX] = {
96     { pre_sp, NULL, MAN_NOTEXT }, /* br */
97     { NULL, NULL, 0 }, /* TH */
98     { pre_SH, post_SH, 0 }, /* SH */
99     { pre_SS, post_SS, 0 }, /* SS */
100    { pre_TP, post_TP, 0 }, /* TP */
101    { pre_PP, NULL, 0 }, /* LP */
102    { pre_PP, NULL, 0 }, /* PP */
103    { pre_PP, NULL, 0 }, /* P */
104    { pre_IP, post_IP, 0 }, /* IP */
105    { pre_HP, post_HP, 0 }, /* HP */
106    { NULL, NULL, 0 }, /* SM */
107    { pre_B, NULL, 0 }, /* SB */
108    { pre_alternate, NULL, 0 }, /* BI */
109    { pre_alternate, NULL, 0 }, /* IB */
110    { pre_alternate, NULL, 0 }, /* BR */
111    { pre_alternate, NULL, 0 }, /* RB */
112    { NULL, NULL, 0 }, /* R */
113    { pre_B, NULL, 0 }, /* B */
114    { pre_I, NULL, 0 }, /* I */
115    { pre_alternate, NULL, 0 }, /* IR */
116    { pre_alternate, NULL, 0 }, /* RI */
117    { pre_ign, NULL, MAN_NOTEXT }, /* na */
118    { pre_sp, NULL, MAN_NOTEXT }, /* sp */
119    { pre_literal, NULL, 0 }, /* nf */
120    { pre_literal, NULL, 0 }, /* fi */
121    { NULL, NULL, 0 }, /* RE */
122    { pre_RS, post_RS, 0 }, /* RS */

```

```

123 { pre_ign, NULL, 0 }, /* DT */
124 { pre_ign, NULL, 0 }, /* UC */
125 { pre_ign, NULL, 0 }, /* PD */
126 { pre_ign, NULL, 0 }, /* AT */
127 { pre_in, NULL, MAN_NOTEXT }, /* in */
128 { pre_ft, NULL, MAN_NOTEXT }, /* ft */
129 { pre_OP, NULL, 0 }, /* OP */
130 };

134 void
135 terminal_man(void *arg, const struct man *man)
136 {
137     struct term *p;
138     const struct man_node *n;
139     const struct man_meta *m;
140     struct mterm mt;

142     p = (struct term *)arg;

144     if (0 == p->defindent)
145         p->defindent = 7;

147     p->overstep = 0;
148     p->maxmargin = p->defmargin;
149     p->tabwidth = term_len(p, 5);

151     if (NULL == p->symtab)
152         p->symtab = mchars_alloc();

154     n = man_node(man);
155     m = man_meta(man);

157     term_begin(p, print_man_head, print_man_foot, m);
158     p->flags |= TERMP_NOSPACE;

160     memset(&mt, 0, sizeof(struct mterm));

162     mt.lmargin[mt.lmargincur] = term_len(p, p->defindent);
163     mt.offset = term_len(p, p->defindent);

165     if (n->child)
166         print_man_nodelist(p, &mt, n->child, m);

168     term_end(p);
169 }

172 static size_t
173 a2height(const struct term *p, const char *cp)
174 {
175     struct roffsu su;

177     if (! a2roffsu(cp, &su, SCALE_VS))
178         SCALE_VS_INIT(&su, atoi(cp));

180     return(term_vspan(p, &su));
181 }

184 static int
185 a2width(const struct term *p, const char *cp)
186 {
187     struct roffsu su;

```

```

189     if (! a2roffsu(cp, &su, SCALE_BU))
190         return(-1);

192     return((int)term_hspan(p, &su));
193 }

195 /*
196  * Printing leading vertical space before a block.
197  * This is used for the paragraph macros.
198  * The rules are pretty simple, since there's very little nesting going
199  * on here. Basically, if we're the first within another block (SS/SH),
200  * then don't emit vertical space. If we are (RS), then do. If not the
201  * first, print it.
202  */
203 static void
204 print_bvspace(struct term *p, const struct man_node *n)
205 {
207     term_newln(p);

209     if (n->body && n->body->child)
210         if (MAN_TBL == n->body->child->type)
211             return;

213     if (MAN_ROOT == n->parent->type || MAN_RS != n->parent->tok)
214         if (NULL == n->prev)
215             return;

217     term_vspace(p);
218 }

220 /* ARGSUSED */
221 static int
222 pre_ign(DECL_ARGS)
223 {
225     return(0);
226 }

229 /* ARGSUSED */
230 static int
231 pre_I(DECL_ARGS)
232 {
234     term_fontrepl(p, TERMFONT_UNDER);
235     return(1);
236 }

239 /* ARGSUSED */
240 static int
241 pre_literal(DECL_ARGS)
242 {
244     term_newln(p);

246     if (MAN_nf == n->tok)
247         mt->fl |= MANT_LITERAL;
248     else
249         mt->fl &= ~MANT_LITERAL;

251     /*
252      * Unlike .IP and .TP, .HP does not have a HEAD.
253      * So in case a second call to term_flushln() is needed,
254      * indentation has to be set up explicitly.

```

```

255  */
256  if (MAN_HP == n->parent->tok && p->rmargin < p->maxrmargin) {
257      p->offset = p->rmargin;
258      p->rmargin = p->maxrmargin;
259      p->flags &= ~(TERMP_NOBREAK | TERMP_TWOSPACE);
260      p->flags |= TERMP_NOSPACE;
261  }
263  return(0);
264 }

266 /* ARGSUSED */
267 static int
268 pre_alternate(DECL_ARGS)
269 {
270     enum termfont      font[2];
271     const struct man_node *nn;
272     int                savelit, i;

274     switch (n->tok) {
275     case (MAN_RB):
276         font[0] = TERMFONT_NONE;
277         font[1] = TERMFONT_BOLD;
278         break;
279     case (MAN_RI):
280         font[0] = TERMFONT_NONE;
281         font[1] = TERMFONT_UNDER;
282         break;
283     case (MAN_BR):
284         font[0] = TERMFONT_BOLD;
285         font[1] = TERMFONT_NONE;
286         break;
287     case (MAN_BI):
288         font[0] = TERMFONT_BOLD;
289         font[1] = TERMFONT_UNDER;
290         break;
291     case (MAN_IR):
292         font[0] = TERMFONT_UNDER;
293         font[1] = TERMFONT_NONE;
294         break;
295     case (MAN_IB):
296         font[0] = TERMFONT_UNDER;
297         font[1] = TERMFONT_BOLD;
298         break;
299     default:
300         abort();
301     }

303     savelit = MANT_LITERAL & mt->fl;
304     mt->fl &= ~MANT_LITERAL;

306     for (i = 0, nn = n->child; nn; nn = nn->next, i = 1 - i) {
307         term_fontrepl(p, font[i]);
308         if (savelit && NULL == nn->next)
309             mt->fl |= MANT_LITERAL;
310         print_man_node(p, mt, nn, m);
311         if (nn->next)
312             p->flags |= TERMP_NOSPACE;
313     }

315     return(0);
316 }

318 /* ARGSUSED */
319 static int
320 pre_B(DECL_ARGS)

```

```

321 {
323     term_fontrepl(p, TERMFONT_BOLD);
324     return(1);
325 }

327 /* ARGSUSED */
328 static int
329 pre_OP(DECL_ARGS)
330 {
332     term_word(p, "[";
333     p->flags |= TERMP_NOSPACE;

335     if (NULL != (n = n->child)) {
336         term_fontrepl(p, TERMFONT_BOLD);
337         term_word(p, n->string);
338     }
339     if (NULL != n && NULL != n->next) {
340         term_fontrepl(p, TERMFONT_UNDER);
341         term_word(p, n->next->string);
342     }

344     term_fontrepl(p, TERMFONT_NONE);
345     p->flags |= TERMP_NOSPACE;
346     term_word(p, "]"");
347     return(0);
348 }

350 /* ARGSUSED */
351 static int
352 pre_ft(DECL_ARGS)
353 {
354     const char      *cp;

356     if (NULL == n->child) {
357         term_fontlast(p);
358         return(0);
359     }

361     cp = n->child->string;
362     switch (*cp) {
363     case ('4'):
364         /* FALLTHROUGH */
365     case ('3'):
366         /* FALLTHROUGH */
367     case ('B'):
368         term_fontrepl(p, TERMFONT_BOLD);
369         break;
370     case ('2'):
371         /* FALLTHROUGH */
372     case ('I'):
373         term_fontrepl(p, TERMFONT_UNDER);
374         break;
375     case ('P'):
376         term_fontlast(p);
377         break;
378     case ('1'):
379         /* FALLTHROUGH */
380     case ('C'):
381         /* FALLTHROUGH */
382     case ('R'):
383         term_fontrepl(p, TERMFONT_NONE);
384         break;
385     default:
386         break;

```

```

387     }
388     return(0);
389 }

391 /* ARGSUSED */
392 static int
393 pre_in(DECL_ARGS)
394 {
395     int            len, less;
396     size_t         v;
397     const char    *cp;

399     term_newln(p);

401     if (NULL == n->child) {
402         p->offset = mt->offset;
403         return(0);
404     }

406     cp = n->child->string;
407     less = 0;

409     if ('-' == *cp)
410         less = -1;
411     else if ('+' == *cp)
412         less = 1;
413     else
414         cp--;

416     if ((len = a2width(p, ++cp)) < 0)
417         return(0);

419     v = (size_t)len;

421     if (less < 0)
422         p->offset -= p->offset > v ? v : p->offset;
423     else if (less > 0)
424         p->offset += v;
425     else
426         p->offset = v;

428     /* Don't let this creep beyond the right margin. */

430     if (p->offset > p->rmargin)
431         p->offset = p->rmargin;

433     return(0);
434 }

437 /* ARGSUSED */
438 static int
439 pre_sp(DECL_ARGS)
440 {
441     size_t        i, len;

443     if ((NULL == n->prev && n->parent)) {
444         if (MAN_SS == n->parent->tok)
445             return(0);
446         if (MAN_SH == n->parent->tok)
447             return(0);
448     }

450     switch (n->tok) {
451     case (MAN_br):
452         len = 0;

```

```

453         break;
454     default:
455         len = n->child ? a2height(p, n->child->string) : 1;
456         break;
457     }

459     if (0 == len)
460         term_newln(p);
461     for (i = 0; i < len; i++)
462         term_vspace(p);

464     return(0);
465 }

468 /* ARGSUSED */
469 static int
470 pre_HP(DECL_ARGS)
471 {
472     size_t        len, one;
473     int           ival;
474     const struct man_node *nn;

476     switch (n->type) {
477     case (MAN_BLOCK):
478         print_bvspace(p, n);
479         return(1);
480     case (MAN_BODY):
481         p->flags |= TERMP_NOBREAK;
482         p->flags |= TERMP_TWOSPACE;
483         break;
484     default:
485         return(0);
486     }

488     len = mt->lmargin[mt->lmargincur];
489     ival = -1;

491     /* Calculate offset. */

493     if (NULL != (nn = n->parent->head->child))
494         if ((ival = a2width(p, nn->string)) >= 0)
495             len = (size_t)ival;

497     one = term_len(p, 1);
498     if (len < one)
499         len = one;

501     p->offset = mt->offset;
502     p->rmargin = mt->offset + len;

504     if (ival >= 0)
505         mt->lmargin[mt->lmargincur] = (size_t)ival;

507     return(1);
508 }

511 /* ARGSUSED */
512 static void
513 post_HP(DECL_ARGS)
514 {

516     switch (n->type) {
517     case (MAN_BLOCK):
518         term_flushln(p);

```



```

519         break;
520     case (MAN_BODY):
521         term_flushln(p);
522         p->flags &= ~TERMP_NOBREAK;
523         p->flags &= ~TERMP_TWOSPACE;
524         p->offset = mt->offset;
525         p->rmargin = p->maxrmargin;
526         break;
527     default:
528         break;
529     }
530 }

533 /* ARGSUSED */
534 static int
535 pre_PP(DECL_ARGS)
536 {
537
538     switch (n->type) {
539     case (MAN_BLOCK):
540         mt->lmargin[mt->lmargincur] = term_len(p, p->defindent);
541         print_bvspace(p, n);
542         break;
543     default:
544         p->offset = mt->offset;
545         break;
546     }

548     return(MAN_HEAD != n->type);
549 }

552 /* ARGSUSED */
553 static int
554 pre_IP(DECL_ARGS)
555 {
556     const struct man_node *nn;
557     size_t len;
558     int savelit, ival;

560     switch (n->type) {
561     case (MAN_BODY):
562         p->flags |= TERMP_NOSPACE;
563         break;
564     case (MAN_HEAD):
565         p->flags |= TERMP_NOBREAK;
566         break;
567     case (MAN_BLOCK):
568         print_bvspace(p, n);
569         /* FALLTHROUGH */
570     default:
571         return(1);
572     }

574     len = mt->lmargin[mt->lmargincur];
575     ival = -1;

577     /* Calculate the offset from the optional second argument. */
578     if (NULL != (nn = n->parent->head->child))
579         if (NULL != (nn = nn->next))
580             if ((ival = a2width(p, nn->string)) >= 0)
581                 len = (size_t)ival;

583     switch (n->type) {
584     case (MAN_HEAD):

```

```

585         /* Handle zero-width lengths. */
586         if (0 == len)
587             len = term_len(p, 1);

589         p->offset = mt->offset;
590         p->rmargin = mt->offset + len;
591         if (ival < 0)
592             break;

594         /* Set the saved left-margin. */
595         mt->lmargin[mt->lmargincur] = (size_t)ival;

597         savelit = MANT_LITERAL & mt->fl;
598         mt->fl &= ~MANT_LITERAL;

600         if (n->child)
601             print_man_node(p, mt, n->child, m);

603         if (savelit)
604             mt->fl |= MANT_LITERAL;

606         return(0);
607     case (MAN_BODY):
608         p->offset = mt->offset + len;
609         p->rmargin = p->maxrmargin;
610         break;
611     default:
612         break;
613     }

615     return(1);
616 }

619 /* ARGSUSED */
620 static void
621 post_IP(DECL_ARGS)
622 {

624     switch (n->type) {
625     case (MAN_HEAD):
626         term_flushln(p);
627         p->flags &= ~TERMP_NOBREAK;
628         p->rmargin = p->maxrmargin;
629         break;
630     case (MAN_BODY):
631         term_newln(p);
632         break;
633     default:
634         break;
635     }
636 }

639 /* ARGSUSED */
640 static int
641 pre_TP(DECL_ARGS)
642 {
643     const struct man_node *nn;
644     size_t len;
645     int savelit, ival;

647     switch (n->type) {
648     case (MAN_HEAD):
649         p->flags |= TERMP_NOBREAK;
650         break;

```

```

651     case (MAN_BODY):
652         p->flags |= TERMP_NOSPACE;
653         break;
654     case (MAN_BLOCK):
655         print_bvspace(p, n);
656         /* FALLTHROUGH */
657     default:
658         return(1);
659 }

661 len = (size_t)mt->lmargin[mt->lmargincur];
662 ival = -1;

664 /* Calculate offset. */

666 if (NULL != (nn = n->parent->head->child))
667     if (nn->string && nn->parent->line == nn->line)
668         if ((ival = a2width(p, nn->string)) >= 0)
669             len = (size_t)ival;

671 switch (n->type) {
672 case (MAN_HEAD):
673     /* Handle zero-length properly. */
674     if (0 == len)
675         len = term_len(p, 1);

677     p->offset = mt->offset;
678     p->rmargin = mt->offset + len;

680     savelit = MANT_LITERAL & mt->fl;
681     mt->fl &= ~MANT_LITERAL;

683     /* Don't print same-line elements. */
684     for (nn = n->child; nn; nn = nn->next)
685         if (nn->line > n->line)
686             print_man_node(p, mt, nn, m);

688     if (savelit)
689         mt->fl |= MANT_LITERAL;
690     if (ival >= 0)
691         mt->lmargin[mt->lmargincur] = (size_t)ival;

693     return(0);
694 case (MAN_BODY):
695     p->offset = mt->offset + len;
696     p->rmargin = p->maxrmargin;
697     break;
698 default:
699     break;
700 }

702 return(1);
703 }

706 /* ARGSUSED */
707 static void
708 post_TP(DECL_ARGS)
709 {
711     switch (n->type) {
712     case (MAN_HEAD):
713         term_flushln(p);
714         p->flags &= ~TERMP_NOBREAK;
715         p->flags &= ~TERMP_TWOSPACE;
716         p->rmargin = p->maxrmargin;

```

```

717         break;
718     case (MAN_BODY):
719         term_newln(p);
720         break;
721     default:
722         break;
723     }
724 }

727 /* ARGSUSED */
728 static int
729 pre_SS(DECL_ARGS)
730 {
732     switch (n->type) {
733     case (MAN_BLOCK):
734         mt->fl &= ~MANT_LITERAL;
735         mt->lmargin[mt->lmargincur] = term_len(p, p->defindent);
736         mt->offset = term_len(p, p->defindent);
737         /* If following a prior empty 'SS', no vspace. */
738         if (n->prev && MAN_SS == n->prev->tok)
739             if (NULL == n->prev->body->child)
740                 break;
741         if (NULL == n->prev)
742             break;
743         term_vspace(p);
744         break;
745     case (MAN_HEAD):
746         term_fontrepl(p, TERMFONT_BOLD);
747         p->offset = term_len(p, p->defindent/2);
748         break;
749     case (MAN_BODY):
750         p->offset = mt->offset;
751         break;
752     default:
753         break;
754     }

756     return(1);
757 }

760 /* ARGSUSED */
761 static void
762 post_SS(DECL_ARGS)
763 {
764     switch (n->type) {
765     case (MAN_HEAD):
766         term_newln(p);
767         break;
768     case (MAN_BODY):
769         term_newln(p);
770         break;
771     default:
772         break;
773     }
774 }
775 }

778 /* ARGSUSED */
779 static int
780 pre_SH(DECL_ARGS)
781 {

```

```

783     switch (n->type) {
784     case (MAN_BLOCK):
785         mt->fl &= ~MANT_LITERAL;
786         mt->lmargin[mt->lmargincur] = term_len(p, p->defindent);
787         mt->offset = term_len(p, p->defindent);
788         /* If following a prior empty 'SH', no vspae. */
789         if (n->prev && MAN_SH == n->prev->tok)
790             if (NULL == n->prev->body->child)
791                 break;
792         /* If the first macro, no vspae. */
793         if (NULL == n->prev)
794             break;
795         term_vspace(p);
796         break;
797     case (MAN_HEAD):
798         term_fontrepl(p, TERMFONT_BOLD);
799         p->offset = 0;
800         break;
801     case (MAN_BODY):
802         p->offset = mt->offset;
803         break;
804     default:
805         break;
806     }
808     return(1);
809 }

```

```

812 /* ARGSUSED */
813 static void
814 post_SH(DECL_ARGS)
815 {
816     switch (n->type) {
817     case (MAN_HEAD):
818         term_newln(p);
819         break;
820     case (MAN_BODY):
821         term_newln(p);
822         break;
823     default:
824         break;
825     }
826 }
827 }

```

```

829 /* ARGSUSED */
830 static int
831 pre_RS(DECL_ARGS)
832 {
833     int         ival;
834     size_t      sz;
836     switch (n->type) {
837     case (MAN_BLOCK):
838         term_newln(p);
839         return(1);
840     case (MAN_HEAD):
841         return(0);
842     default:
843         break;
844     }
846     sz = term_len(p, p->defindent);
848     if (NULL != (n = n->parent->head->child))

```

```

849         if ((ival = a2width(p, n->string)) >= 0)
850             sz = (size_t)ival;
852     mt->offset += sz;
853     p->rmargin = p->maxrmargin;
854     p->offset = mt->offset < p->rmargin ? mt->offset : p->rmargin;
856     if (++mt->lmarginsz < MAXMARGINS)
857         mt->lmargincur = mt->lmarginsz;
859     mt->lmargin[mt->lmargincur] = mt->lmargin[mt->lmargincur - 1];
860     return(1);
861 }

```

```

863 /* ARGSUSED */
864 static void
865 post_RS(DECL_ARGS)
866 {
867     int         ival;
868     size_t      sz;
870     switch (n->type) {
871     case (MAN_BLOCK):
872         return;
873     case (MAN_HEAD):
874         return;
875     default:
876         term_newln(p);
877         break;
878     }

```

```

880     sz = term_len(p, p->defindent);
882     if (NULL != (n = n->parent->head->child))
883         if ((ival = a2width(p, n->string)) >= 0)
884             sz = (size_t)ival;
886     mt->offset = mt->offset < sz ? 0 : mt->offset - sz;
887     p->offset = mt->offset;
889     if (--mt->lmarginsz < MAXMARGINS)
890         mt->lmargincur = mt->lmarginsz;
891 }

```

```

893 static void
894 print_man_node(DECL_ARGS)
895 {
896     size_t      rm, rmax;
897     int         c;
899     switch (n->type) {
900     case (MAN_TEXT):
901         /*
902          * If we have a blank line, output a vertical space.
903          * If we have a space as the first character, break
904          * before printing the line's data.
905          */
906         if ('\0' == *n->string) {
907             term_vspace(p);
908             return;
909         } else if (' ' == *n->string && MAN_LINE & n->flags)
910             term_newln(p);
912         term_word(p, n->string);
914         /*

```

```

915     * If we're in a literal context, make sure that words
916     * together on the same line stay together. This is a
917     * POST-printing call, so we check the NEXT word. Since
918     * -man doesn't have nested macros, we don't need to be
919     * more specific than this.
920     */
921     if (MANT_LITERAL & mt->fl && ! (TERMP_NOBREAK & p->flags) &&
922         (NULL == n->next ||
923          n->next->line > n->line)) {
924         rm = p->rmargin;
925         rmax = p->maxrmargin;
926         p->rmargin = p->maxrmargin = TERM_MAXMARGIN;
927         p->flags |= TERMP_NOSPACE;
928         term_flushln(p);
929         p->rmargin = rm;
930         p->maxrmargin = rmax;
931     }

933     if (MAN_EOS & n->flags)
934         p->flags |= TERMP_SENTENCE;
935     return;
936 case (MAN_EQN):
937     term_eqn(p, n->eqn);
938     return;
939 case (MAN_TBL):
940     /*
941     * Tables are preceded by a newline. Then process a
942     * table line, which will cause line termination,
943     */
944     if (TBL_SPAN_FIRST & n->span->flags)
945         term_newln(p);
946     term_tbl(p, n->span);
947     return;
948 default:
949     break;
950 }

952 if ( ! (MAN_NOTEXT & termacts[n->tok].flags))
953     term_fontrpl(p, TERMFONT_NONE);

955 c = 1;
956 if (termacts[n->tok].pre)
957     c = (*termacts[n->tok].pre)(p, mt, n, m);

959 if (c && n->child)
960     print_man_nodelist(p, mt, n->child, m);

962 if (termacts[n->tok].post)
963     (*termacts[n->tok].post)(p, mt, n, m);
964 if ( ! (MAN_NOTEXT & termacts[n->tok].flags))
965     term_fontrpl(p, TERMFONT_NONE);

967 if (MAN_EOS & n->flags)
968     p->flags |= TERMP_SENTENCE;
969 }

972 static void
973 print_man_nodelist(DECL_ARGS)
974 {

976     print_man_node(p, mt, n, m);
977     if ( ! n->next)
978         return;
979     print_man_nodelist(p, mt, n->next, m);
980 }

```

```

983 static void
984 print_man_foot(struct term *p, const void *arg)
985 {
986     char          title[BUFSIZ];
987     size_t        datelen;
988     const struct man_meta *meta;

990     meta = (const struct man_meta *)arg;
991     assert(meta->title);
992     assert(meta->msec);
993     assert(meta->date);

995     term_fontrpl(p, TERMFONT_NONE);

997     term_vspace(p);

999     /*
1000     * Temporary, undocumented option to imitate mdoc(7) output.
1001     * In the bottom right corner, use the source instead of
1002     * the title.
1003     */

1005     if ( ! p->mdocstyle) {
1006         term_vspace(p);
1007         term_vspace(p);
1008         snprintf(title, BUFSIZ, "%s(%s)", meta->title, meta->msec);
1009     } else if (meta->source) {
1010         strncpy(title, meta->source, BUFSIZ);
1011     } else {
1012         title[0] = '\0';
1013     }
1014     datelen = term_strlen(p, meta->date);

1016     /* Bottom left corner: manual source. */

1018     p->flags |= TERMP_NOSPACE | TERMP_NOBREAK;
1019     p->offset = 0;
1020     p->rmargin = (p->maxrmargin - datelen + term_len(p, 1)) / 2;

1022     if (meta->source)
1023         term_word(p, meta->source);
1024     term_flushln(p);

1026     /* At the bottom in the middle: manual date. */

1028     p->flags |= TERMP_NOSPACE;
1029     p->offset = p->rmargin;
1030     p->rmargin = p->maxrmargin - term_strlen(p, title);
1031     if (p->offset + datelen >= p->rmargin)
1032         p->rmargin = p->offset + datelen;

1034     term_word(p, meta->date);
1035     term_flushln(p);

1037     /* Bottom right corner: manual title and section. */

1039     p->flags &= ~TERMP_NOBREAK;
1040     p->flags |= TERMP_NOSPACE;
1041     p->offset = p->rmargin;
1042     p->rmargin = p->maxrmargin;

1044     term_word(p, title);
1045     term_flushln(p);
1046 }

```

```

1049 static void
1050 print_man_head(struct term *p, const void *arg)
1051 {
1052     char          buf[BUFSIZ], title[BUFSIZ];
1053     size_t        buflen, titlen;
1054     const struct man_meta *m;
1055
1056     m = (const struct man_meta *)arg;
1057     assert(m->title);
1058     assert(m->msec);
1059
1060     if (m->vol)
1061         strcpy(buf, m->vol, BUFSIZ);
1062     else
1063         buf[0] = '\0';
1064     buflen = term_strlen(p, buf);
1065
1066     /* Top left corner: manual title and section. */
1067
1068     snprintf(title, BUFSIZ, "%s(%s)", m->title, m->msec);
1069     titlen = term_strlen(p, title);
1070
1071     p->flags |= TERMP_NOBREAK | TERMP_NOSPACE;
1072     p->offset = 0;
1073     p->rmargin = 2 * (titlen+1) + buflen < p->maxrmargin ?
1074         (p->maxrmargin -
1075          term_strlen(p, buf) + term_len(p, 1)) / 2 :
1076         p->maxrmargin - buflen;
1077
1078     term_word(p, title);
1079     term_flushln(p);
1080
1081     /* At the top in the middle: manual volume. */
1082
1083     p->flags |= TERMP_NOSPACE;
1084     p->offset = p->rmargin;
1085     p->rmargin = p->offset + buflen + titlen < p->maxrmargin ?
1086         p->maxrmargin - titlen : p->maxrmargin;
1087
1088     term_word(p, buf);
1089     term_flushln(p);
1090
1091     /* Top right corner: title and section, again. */
1092
1093     p->flags &= ~TERMP_NOBREAK;
1094     if (p->rmargin + titlen <= p->maxrmargin) {
1095         p->flags |= TERMP_NOSPACE;
1096         p->offset = p->rmargin;
1097         p->rmargin = p->maxrmargin;
1098         term_word(p, title);
1099         term_flushln(p);
1100     }
1101
1102     p->flags &= ~TERMP_NOSPACE;
1103     p->offset = 0;
1104     p->rmargin = p->maxrmargin;
1105
1106     /*
1107      * Groff prints three blank lines before the content.
1108      * Do the same, except in the temporary, undocumented
1109      * mode imitating mdoc(7) output.
1110      */
1111
1112     term_vspace(p);

```

```

1113         if ( ! p->mdocstyle) {
1114             term_vspace(p);
1115             term_vspace(p);
1116         }
1117     }

```

```

*****
11290 Sat Jul 19 19:24:30 2014
new/usr/src/cmd/mandoc/man_validate.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: man_validate.c,v 1.80 2012/01/03 15:16:24 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2010 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifndef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <sys/types.h>
23
24 #include <assert.h>
25 #include <ctype.h>
26 #include <errno.h>
27 #include <limits.h>
28 #include <stdarg.h>
29 #include <stdlib.h>
30 #include <string.h>
31 #include <time.h>
32
33 #include "man.h"
34 #include "mandoc.h"
35 #include "libman.h"
36 #include "libmandoc.h"
37
38 #define CHKARGS    struct man *m, struct man_node *n
39
40 typedef int      (*v_check)(CHKARGS);
41
42 struct man_valid {
43     v_check  *pres;
44     v_check  *posts;
45 };
46
47 static int      check_eq0(CHKARGS);
48 static int      check_eq2(CHKARGS);
49 static int      check_le1(CHKARGS);
50 static int      check_ge2(CHKARGS);
51 static int      check_le5(CHKARGS);
52 static int      check_par(CHKARGS);
53 static int      check_part(CHKARGS);
54 static int      check_root(CHKARGS);
55 static void      check_text(CHKARGS);

```

```

57 static int      post_AT(CHKARGS);
58 static int      post_vs(CHKARGS);
59 static int      post_fi(CHKARGS);
60 static int      post_ft(CHKARGS);
61 static int      post_nf(CHKARGS);
62 static int      post_sec(CHKARGS);
63 static int      post_TH(CHKARGS);
64 static int      post_UC(CHKARGS);
65 static int      pre_sec(CHKARGS);
66
67 static v_check  posts_at[] = { post_AT, NULL };
68 static v_check  posts_br[] = { post_vs, check_eq0, NULL };
69 static v_check  posts_eq0[] = { check_eq0, NULL };
70 static v_check  posts_eq2[] = { check_eq2, NULL };
71 static v_check  posts_fi[] = { check_eq0, post_fi, NULL };
72 static v_check  posts_ft[] = { post_ft, NULL };
73 static v_check  posts_nf[] = { check_eq0, post_nf, NULL };
74 static v_check  posts_par[] = { check_par, NULL };
75 static v_check  posts_part[] = { check_part, NULL };
76 static v_check  posts_sec[] = { post_sec, NULL };
77 static v_check  posts_sp[] = { post_vs, check_le1, NULL };
78 static v_check  posts_th[] = { check_ge2, check_le5, post_TH, NULL };
79 static v_check  posts_uc[] = { post_UC, NULL };
80 static v_check  pres_sec[] = { pre_sec, NULL };
81
82 static const struct man_valid man_valids[MAN_MAX] = {
83     { NULL, posts_br }, /* br */
84     { NULL, posts_th }, /* TH */
85     { pres_sec, posts_sec }, /* SH */
86     { pres_sec, posts_sec }, /* SS */
87     { NULL, NULL }, /* TP */
88     { NULL, posts_par }, /* LP */
89     { NULL, posts_par }, /* PP */
90     { NULL, posts_par }, /* P */
91     { NULL, NULL }, /* IP */
92     { NULL, NULL }, /* HP */
93     { NULL, NULL }, /* SM */
94     { NULL, NULL }, /* SB */
95     { NULL, NULL }, /* BI */
96     { NULL, NULL }, /* IB */
97     { NULL, NULL }, /* BR */
98     { NULL, NULL }, /* RB */
99     { NULL, NULL }, /* R */
100    { NULL, NULL }, /* B */
101    { NULL, NULL }, /* I */
102    { NULL, NULL }, /* IR */
103    { NULL, NULL }, /* RI */
104    { NULL, posts_eq0 }, /* na */
105    { NULL, posts_sp }, /* sp */
106    { NULL, posts_nf }, /* nf */
107    { NULL, posts_fi }, /* fi */
108    { NULL, NULL }, /* RE */
109    { NULL, posts_part }, /* RS */
110    { NULL, NULL }, /* DT */
111    { NULL, posts_uc }, /* UC */
112    { NULL, NULL }, /* PD */
113    { NULL, posts_at }, /* AT */
114    { NULL, NULL }, /* in */
115    { NULL, posts_ft }, /* ft */
116    { NULL, posts_eq2 }, /* OP */
117 };
118
119
120 int
121 man_valid_pre(struct man *m, struct man_node *n)
122 {

```

```

123     v_check      *cp;

125     switch (n->type) {
126     case (MAN_TEXT):
127         /* FALLTHROUGH */
128     case (MAN_ROOT):
129         /* FALLTHROUGH */
130     case (MAN_EQN):
131         /* FALLTHROUGH */
132     case (MAN_TBL):
133         return(1);
134     default:
135         break;
136     }

138     if (NULL == (cp = man_valids[n->tok].pres))
139         return(1);
140     for ( ; *cp; cp++)
141         if ( ! (*cp)(m, n))
142             return(0);
143     return(1);
144 }

147 int
148 man_valid_post(struct man *m)
149 {
150     v_check      *cp;

152     if (MAN_VALID & m->last->flags)
153         return(1);
154     m->last->flags |= MAN_VALID;

156     switch (m->last->type) {
157     case (MAN_TEXT):
158         check_text(m, m->last);
159         return(1);
160     case (MAN_ROOT):
161         return(check_root(m, m->last));
162     case (MAN_EQN):
163         /* FALLTHROUGH */
164     case (MAN_TBL):
165         return(1);
166     default:
167         break;
168     }

170     if (NULL == (cp = man_valids[m->last->tok].posts))
171         return(1);
172     for ( ; *cp; cp++)
173         if ( ! (*cp)(m, m->last))
174             return(0);

176     return(1);
177 }

180 static int
181 check_root(CHKARGS)
182 {

184     if (MAN_BLINE & m->flags)
185         man_nmsg(m, n, MANDOCERR_SCOPEEXIT);
186     else if (MAN_ELINE & m->flags)
187         man_nmsg(m, n, MANDOCERR_SCOPEEXIT);

```

```

189     m->flags &= ~MAN_BLINE;
190     m->flags &= ~MAN_ELINE;

192     if (NULL == m->first->child) {
193         man_nmsg(m, n, MANDOCERR_NODOCBODY);
194         return(0);
195     } else if (NULL == m->meta.title) {
196         man_nmsg(m, n, MANDOCERR_NOTITLE);

198         /*
199          * If a title hasn't been set, do so now (by
200          * implication, date and section also aren't set).
201          */

203         m->meta.title = mandoc_strdup("unknown");
204         m->meta.msec = mandoc_strdup("");
205         m->meta.date = mandoc_normdate
206             (m->parse, NULL, n->line, n->pos);
207     }

209     return(1);
210 }

212 static void
213 check_text(CHKARGS)
214 {
215     char          *cp, *p;

217     if (MAN_LITERAL & m->flags)
218         return;

220     cp = n->string;
221     for (p = cp; NULL != (p = strchr(p, '\t')); p++)
222         man_pmsg(m, n->line, (int)(p - cp), MANDOCERR_BADTAB);
223 }

225 #define INEQ_DEFINE(x, ineq, name) \
226 static int \
227 check_##name(CHKARGS) \
228 { \
229     if (n->nchild ineq (x)) \
230         return(1); \
231     mandoc_vmsg(MANDOCERR_ARGCOUNT, m->parse, n->line, n->pos, \
232         "line arguments %s %d (have %d)", \
233         #ineq, (x), n->nchild); \
234     return(1); \
235 }

237 INEQ_DEFINE(0, ==, eq0)
238 INEQ_DEFINE(2, ==, eq2)
239 INEQ_DEFINE(1, <=, le1)
240 INEQ_DEFINE(2, >=, ge2)
241 INEQ_DEFINE(5, <=, le5)

243 static int
244 post_ft(CHKARGS)
245 {
246     char          *cp;
247     int           ok;

249     if (0 == n->nchild)
250         return(1);

252     ok = 0;
253     cp = n->child->string;
254     switch (*cp) {

```



```

387         break;
388     }
389     }
390     m->meta.title = mandoc_strdup(n->string);
391 } else
392     m->meta.title = mandoc_strdup("");
393
394 /* TITLE ->MSEC<- DATE SOURCE VOL */
395
396 if (n)
397     n = n->next;
398 if (n && n->string)
399     m->meta.msec = mandoc_strdup(n->string);
400 else
401     m->meta.msec = mandoc_strdup("");
402
403 /* TITLE MSEC ->DATE<- SOURCE VOL */
404
405 if (n)
406     n = n->next;
407 if (n && n->string && '\0' != n->string[0]) {
408     pos = n->pos;
409     m->meta.date = mandoc_normdate
410         (m->parse, n->string, line, pos);
411 } else
412     m->meta.date = mandoc_strdup("");
413
414 /* TITLE MSEC DATE ->SOURCE<- VOL */
415
416 if (n && (n = n->next))
417     m->meta.source = mandoc_strdup(n->string);
418
419 /* TITLE MSEC DATE SOURCE ->VOL<- */
420 /* If missing, use the default VOL name for MSEC. */
421
422 if (n && (n = n->next))
423     m->meta.vol = mandoc_strdup(n->string);
424 else if ('\0' != m->meta.msec[0] &&
425         (NULL != (p = mandoc_a2msec(m->meta.msec))))
426     m->meta.vol = mandoc_strdup(p);
427
428 /*
429  * Remove the 'TH' node after we've processed it for our
430  * meta-data.
431  */
432 man_node_delete(m, m->last);
433 return(1);
434 }
435
436 static int
437 post_nf(CHKARGS)
438 {
439     if (MAN_LITERAL & m->flags)
440         man_nmsg(m, n, MANDOCERR_SCOPEREPI);
441
442     m->flags |= MAN_LITERAL;
443     return(1);
444 }
445
446 static int
447 post_fi(CHKARGS)
448 {
449     if (! (MAN_LITERAL & m->flags))
450         man_nmsg(m, n, MANDOCERR_WNOSCOPE);

```

```

454     m->flags &= ~MAN_LITERAL;
455     return(1);
456 }
457
458 static int
459 post_UC(CHKARGS)
460 {
461     static const char * const bsd_versions[] = {
462         "3rd Berkeley Distribution",
463         "4th Berkeley Distribution",
464         "4.2 Berkeley Distribution",
465         "4.3 Berkeley Distribution",
466         "4.4 Berkeley Distribution",
467     };
468
469     const char *p, *s;
470
471     n = n->child;
472
473     if (NULL == n || MAN_TEXT != n->type)
474         p = bsd_versions[0];
475     else {
476         s = n->string;
477         if (0 == strcmp(s, "3"))
478             p = bsd_versions[0];
479         else if (0 == strcmp(s, "4"))
480             p = bsd_versions[1];
481         else if (0 == strcmp(s, "5"))
482             p = bsd_versions[2];
483         else if (0 == strcmp(s, "6"))
484             p = bsd_versions[3];
485         else if (0 == strcmp(s, "7"))
486             p = bsd_versions[4];
487         else
488             p = bsd_versions[0];
489     }
490
491     if (m->meta.source)
492         free(m->meta.source);
493
494     m->meta.source = mandoc_strdup(p);
495     return(1);
496 }
497
498 static int
499 post_AT(CHKARGS)
500 {
501     static const char * const unix_versions[] = {
502         "7th Edition",
503         "System III",
504         "System V",
505         "System V Release 2",
506     };
507
508     const char *p, *s;
509     struct man_node *nn;
510
511     n = n->child;
512
513     if (NULL == n || MAN_TEXT != n->type)
514         p = unix_versions[0];
515     else {
516         s = n->string;
517         if (0 == strcmp(s, "3"))
518             p = unix_versions[0];

```

```
519         else if (0 == strcmp(s, "4"))
520             p = unix_versions[1];
521         else if (0 == strcmp(s, "5")) {
522             nn = n->next;
523             if (nn && MAN_TEXT == nn->type && nn->string[0])
524                 p = unix_versions[3];
525             else
526                 p = unix_versions[2];
527         } else
528             p = unix_versions[0];
529     }

531     if (m->meta.source)
532         free(m->meta.source);

534     m->meta.source = mandoc_strdup(p);
535     return(1);
536 }

538 static int
539 post_vs(CHKARGS)
540 {
542     /*
543      * Don't warn about this because it occurs in pod2man and would
544      * cause considerable (unfixable) warnage.
545      */
546     if (NULL == n->prev && MAN_ROOT == n->parent->type)
547         man_node_delete(m, n);

549     return(1);
550 }
```

```

*****
14130 Sat Jul 19 19:24:30 2014
new/usr/src/cmd/mandoc/mandoc.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: mandoc.c,v 1.62 2011/12/03 16:08:51 schwarze Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHORS DISCLAIM ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <sys/types.h>
23
24 #include <assert.h>
25 #include <ctype.h>
26 #include <errno.h>
27 #include <limits.h>
28 #include <stdlib.h>
29 #include <stdio.h>
30 #include <string.h>
31 #include <time.h>
32
33 #include "mandoc.h"
34 #include "libmandoc.h"
35
36 #define DATESIZE 32
37
38 static int      a2time(time_t *, const char *, const char *);
39 static char     *time2a(time_t);
40 static int      numescape(const char *);
41
42 /*
43  * Pass over recursive numerical expressions. This context of this
44  * function is important: it's only called within character-terminating
45  * escapes (e.g., \s[xxxxyy]), so all we need to do is handle initial
46  * recursion: we don't care about what's in these blocks.
47  * This returns the number of characters skipped or -1 if an error
48  * occurs (the caller should bail).
49  */
50 static int
51 numescape(const char *start)
52 {
53     int          i;
54     size_t       sz;
55     const char   *cp;

```

```

57     i = 0;
58
59     /* The expression consists of a subexpression. */
60
61     if ('\\"' == start[i]) {
62         cp = &start[++i];
63         /*
64          * Read past the end of the subexpression.
65          * Bail immediately on errors.
66          */
67         if (ESCAPE_ERROR == mandoc_escape(&cp, NULL, NULL))
68             return(-1);
69         return(i + cp - &start[i]);
70     }
71
72     if ((' ' != start[i++])
73         return(0);
74
75     /*
76      * A parenthesised subexpression. Read until the closing
77      * parenthesis, making sure to handle any nested subexpressions
78      * that might ruin our parse.
79      */
80
81     while (')' != start[i]) {
82         sz = strcspn(&start[i], "\\\"");
83         i += (int)sz;
84
85         if ('\0' == start[i])
86             return(-1);
87         else if ('\\"' != start[i])
88             continue;
89
90         cp = &start[++i];
91         if (ESCAPE_ERROR == mandoc_escape(&cp, NULL, NULL))
92             return(-1);
93         i += cp - &start[i];
94     }
95
96     /* Read past the terminating ')'. */
97     return(++i);
98 }
99
100 enum mandoc_esc
101 mandoc_escape(const char **end, const char **start, int *sz)
102 {
103     char          c, term, numeric;
104     int           i, lim, ssize, rlim;
105     const char    *cp, *rstart;
106     enum mandoc_esc gly;
107
108     cp = *end;
109     rstart = cp;
110     if (start)
111         *start = rstart;
112     i = lim = 0;
113     gly = ESCAPE_ERROR;
114     term = numeric = '\0';
115
116     switch ((c = cp[i++])) {
117     /*
118      * First the glyphs. There are several different forms of
119      * these, but each eventually returns a substring of the glyph
120      * name.
121      */
122     case ((' ':

```

```

123     gly = ESCAPE_SPECIAL;
124     lim = 2;
125     break;
126 case ('['):
127     gly = ESCAPE_SPECIAL;
128     /*
129     * Unicode escapes are defined in groff as [\uXXXX] to
130     * [\u0FFFF], where the contained value must be a valid
131     * Unicode codepoint. Here, however, only check whether
132     * it's not a zero-width escape.
133     */
134     if ('u' == cp[i] && ']' != cp[i + 1])
135         gly = ESCAPE_UNICODE;
136     term = ']';
137     break;
138 case ('C'):
139     if ('\'' != cp[i])
140         return(ESCAPE_ERROR);
141     gly = ESCAPE_SPECIAL;
142     term = '\'';
143     break;
144
145 /*
146 * Handle all triggers matching \X(xy, \Xx, and \X[xxxx], where
147 * 'X' is the trigger. These have opaque sub-strings.
148 */
149 case ('F'):
150     /* FALLTHROUGH */
151 case ('g'):
152     /* FALLTHROUGH */
153 case ('k'):
154     /* FALLTHROUGH */
155 case ('M'):
156     /* FALLTHROUGH */
157 case ('m'):
158     /* FALLTHROUGH */
159 case ('n'):
160     /* FALLTHROUGH */
161 case ('V'):
162     /* FALLTHROUGH */
163 case ('Y'):
164     gly = ESCAPE_IGNORE;
165     /* FALLTHROUGH */
166 case ('f'):
167     if (ESCAPE_ERROR == gly)
168         gly = ESCAPE_FONT;
169
170     rstart = &cp[i];
171     if (start)
172         *start = rstart;
173
174     switch (cp[i++]) {
175     case ('('):
176         lim = 2;
177         break;
178     case ('['):
179         term = ']';
180         break;
181     default:
182         lim = 1;
183         i--;
184         break;
185     }
186     break;
187
188 /*

```

```

189     * These escapes are of the form \X'Y', where 'X' is the trigger
190     * and 'Y' is any string. These have opaque sub-strings.
191     */
192 case ('A'):
193     /* FALLTHROUGH */
194 case ('b'):
195     /* FALLTHROUGH */
196 case ('D'):
197     /* FALLTHROUGH */
198 case ('o'):
199     /* FALLTHROUGH */
200 case ('R'):
201     /* FALLTHROUGH */
202 case ('X'):
203     /* FALLTHROUGH */
204 case ('Z'):
205     if ('\'' != cp[i++])
206         return(ESCAPE_ERROR);
207     gly = ESCAPE_IGNORE;
208     term = '\'';
209     break;
210
211 /*
212 * These escapes are of the form \X'N', where 'X' is the trigger
213 * and 'N' resolves to a numerical expression.
214 */
215 case ('B'):
216     /* FALLTHROUGH */
217 case ('h'):
218     /* FALLTHROUGH */
219 case ('H'):
220     /* FALLTHROUGH */
221 case ('L'):
222     /* FALLTHROUGH */
223 case ('l'):
224     gly = ESCAPE_NUMBERED;
225     /* FALLTHROUGH */
226 case ('S'):
227     /* FALLTHROUGH */
228 case ('v'):
229     /* FALLTHROUGH */
230 case ('w'):
231     /* FALLTHROUGH */
232 case ('x'):
233     if (ESCAPE_ERROR == gly)
234         gly = ESCAPE_IGNORE;
235     if ('\'' != cp[i++])
236         return(ESCAPE_ERROR);
237     term = numeric = '\'';
238     break;
239
240 /*
241 * Special handling for the numbered character escape.
242 * XXX Do any other escapes need similar handling?
243 */
244 case ('N'):
245     if ('\0' == cp[i])
246         return(ESCAPE_ERROR);
247     *end = &cp[i+1];
248     if (isdigit((unsigned char)cp[i-1]))
249         return(ESCAPE_IGNORE);
250     while (isdigit((unsigned char)**end))
251         (*end)++;
252     if (start)
253         *start = &cp[i];
254     if (sz)

```

```

255         *sz = *end - &cp[i];
256         if ('\0' != **end)
257             (*end)++;
258         return(ESCAPE_NUMBERED);

260 /*
261  * Sizes get a special category of their own.
262  */
263 case ('s'):
264     gly = ESCAPE_IGNORE;

266     rstart = &cp[i];
267     if (start)
268         *start = rstart;

270 /* See +/- counts as a sign. */
271 c = cp[i];
272 if ('+' == c || '-' == c || ASCII_HYPH == c)
273     ++i;

275 switch (cp[i++]) {
276 case ('('):
277     lim = 2;
278     break;
279 case ('['):
280     term = numeric = ']';
281     break;
282 case ('\'''):
283     term = numeric = '`';
284     break;
285 default:
286     lim = 1;
287     i--;
288     break;
289 }

291 /* See +/- counts as a sign. */
292 c = cp[i];
293 if ('+' == c || '-' == c || ASCII_HYPH == c)
294     ++i;

296 break;

298 /*
299  * Anything else is assumed to be a glyph.
300  */
301 default:
302     gly = ESCAPE_SPECIAL;
303     lim = 1;
304     i--;
305     break;
306 }

308 assert(ESCAPE_ERROR != gly);

310 rstart = &cp[i];
311 if (start)
312     *start = rstart;

314 /*
315  * If a terminating block has been specified, we need to
316  * handle the case of recursion, which could have their
317  * own terminating blocks that mess up our parse. This, by the
318  * way, means that the "start" and "size" values will be
319  * effectively meaningless.
320  */

```

```

322     ssz = 0;
323     if (numeric && -1 == (ssz = numescape(&cp[i])))
324         return(ESCAPE_ERROR);

326     i += ssz;
327     rlim = -1;

329 /*
330  * We have a character terminator. Try to read up to that
331  * character. If we can't (i.e., we hit the nil), then return
332  * an error; if we can, calculate our length, read past the
333  * terminating character, and exit.
334  */

336     if ('\0' != term) {
337         *end = strchr(&cp[i], term);
338         if ('\0' == *end)
339             return(ESCAPE_ERROR);

341         rlim = *end - &cp[i];
342         if (sz)
343             *sz = rlim;
344         (*end)++;
345         goto out;
346     }

348     assert(lim > 0);

350 /*
351  * We have a numeric limit. If the string is shorter than that,
352  * stop and return an error. Else adjust our endpoint, length,
353  * and return the current glyph.
354  */

356     if ((size_t)lim > strlen(&cp[i]))
357         return(ESCAPE_ERROR);

359     rlim = lim;
360     if (sz)
361         *sz = rlim;

363     *end = &cp[i] + lim;

365 out:
366     assert(rlim >= 0 && rstart);

368 /* Run post-processors. */

370     switch (gly) {
371     case (ESCAPE_FONT):
372         /*
373          * Pretend that the constant-width font modes are the
374          * same as the regular font modes.
375          */
376         if (2 == rlim && 'C' == *rstart)
377             rstart++;
378         else if (1 != rlim)
379             break;

381         switch (*rstart) {
382         case ('3'):
383             /* FALLTHROUGH */
384         case ('B'):
385             gly = ESCAPE_FONTBOLD;
386             break;

```

```

387         case ('2'):
388             /* FALLTHROUGH */
389         case ('I'):
390             gly = ESCAPE_FONTITALIC;
391             break;
392         case ('P'):
393             gly = ESCAPE_FONTPREV;
394             break;
395         case ('l'):
396             /* FALLTHROUGH */
397         case ('R'):
398             gly = ESCAPE_FONTROMAN;
399             break;
400     }
401     break;
402     case (ESCAPE_SPECIAL):
403         if (l != rlim)
404             break;
405         if ('c' == *rstart)
406             gly = ESCAPE_NOSPACE;
407         break;
408     default:
409         break;
410 }
411
412 return(gly);
413 }
414
415 void *
416 mandoc_calloc(size_t num, size_t size)
417 {
418     void *ptr;
419
420     ptr = calloc(num, size);
421     if (NULL == ptr) {
422         perror(NULL);
423         exit((int)MANDOCLEVEL_SYSERR);
424     }
425
426     return(ptr);
427 }
428
429 void *
430 mandoc_malloc(size_t size)
431 {
432     void *ptr;
433
434     ptr = malloc(size);
435     if (NULL == ptr) {
436         perror(NULL);
437         exit((int)MANDOCLEVEL_SYSERR);
438     }
439
440     return(ptr);
441 }
442
443 void *
444 mandoc_realloc(void *ptr, size_t size)
445 {
446     ptr = realloc(ptr, size);
447     if (NULL == ptr) {
448         perror(NULL);
449         exit((int)MANDOCLEVEL_SYSERR);

```

```

453     }
454
455     return(ptr);
456 }
457
458 char *
459 mandoc_strndup(const char *ptr, size_t sz)
460 {
461     char *p;
462
463     p = mandoc_malloc(sz + 1);
464     memcpy(p, ptr, sz);
465     p[(int)sz] = '\0';
466     return(p);
467 }
468
469 char *
470 mandoc_strdup(const char *ptr)
471 {
472     char *p;
473
474     p = strdup(ptr);
475     if (NULL == p) {
476         perror(NULL);
477         exit((int)MANDOCLEVEL_SYSERR);
478     }
479
480     return(p);
481 }
482
483 /*
484  * Parse a quoted or unquoted roff-style request or macro argument.
485  * Return a pointer to the parsed argument, which is either the original
486  * pointer or advanced by one byte in case the argument is quoted.
487  * Null-terminate the argument in place.
488  * Collapse pairs of quotes inside quoted arguments.
489  * Advance the argument pointer to the next argument,
490  * or to the null byte terminating the argument line.
491  */
492 char *
493 mandoc_getarg(struct mparse *parse, char **cpp, int ln, int *pos)
494 {
495     char *start, *cp;
496     int quoted, pairs, white;
497
498     /* Quoting can only start with a new word. */
499     start = *cpp;
500     quoted = 0;
501     if ('"' == *start) {
502         quoted = 1;
503         start++;
504     }
505
506     pairs = 0;
507     white = 0;
508     for (cp = start; '\0' != *cp; cp++) {
509         /* Move left after quoted quotes and escaped backslashes. */
510         if (pairs)
511             cp[-pairs] = cp[0];
512         if ('\\' == cp[0]) {
513             if ('\\' == cp[1]) {
514                 /* Poor man's copy mode. */
515                 pairs++;
516                 cp++;
517             } else if (0 == quoted && ' ' == cp[1])
518                 /* Skip escaped blanks. */

```

```

519         cp++;
520     } else if (0 == quoted) {
521         if (' ' == cp[0]) {
522             /* Unescaped blanks end unquoted args. */
523             white = 1;
524             break;
525         }
526     } else if ('"' == cp[0]) {
527         if ('"' == cp[1]) {
528             /* Quoted quotes collapse. */
529             pairs++;
530             cp++;
531         } else {
532             /* Unquoted quotes end quoted args. */
533             quoted = 2;
534             break;
535         }
536     }
537 }

539 /* Quoted argument without a closing quote. */
540 if (1 == quoted)
541     mandoc_msg(MANDOCERR_BADQUOTE, parse, ln, *pos, NULL);

543 /* Null-terminate this argument and move to the next one. */
544 if (pairs)
545     cp[-pairs] = '\0';
546 if ('\0' != *cp) {
547     *cp++ = '\0';
548     while (' ' == *cp)
549         cp++;
550 }
551 *pos += (int)(cp - start) + (quoted ? 1 : 0);
552 *cpp = cp;

554 if ('\0' == *cp && (white || ' ' == cp[-1]))
555     mandoc_msg(MANDOCERR_EOLNSPACE, parse, ln, *pos, NULL);

557 return(start);
558 }

560 static int
561 a2time(time_t *t, const char *fmt, const char *p)
562 {
563     struct tm      tm;
564     char           *pp;

566     memset(&tm, 0, sizeof(struct tm));

568     pp = NULL;
569 #ifdef HAVE_STRPTIME
570     pp = strptime(p, fmt, &tm);
571 #endif
572     if (NULL != pp && '\0' == *pp) {
573         *t = mktime(&tm);
574         return(1);
575     }

577     return(0);
578 }

580 static char *
581 time2a(time_t t)
582 {
583     struct tm      *tm;
584     char           *buf, *p;

```

```

585     size_t      ssz;
586     int         isz;

588     tm = localtime(&t);

590     /*
591     * Reserve space:
592     * up to 9 characters for the month (September) + blank
593     * up to 2 characters for the day + comma + blank
594     * 4 characters for the year and a terminating '\0'
595     */
596     p = buf = mandoc_malloc(10 + 4 + 4 + 1);

598     if (0 == (ssz = strftime(p, 10 + 1, "%B ", tm)))
599         goto fail;
600     p += (int)ssz;

602     if (-1 == (isz = snprintf(p, 4 + 1, "%d", tm->tm_mday)))
603         goto fail;
604     p += isz;

606     if (0 == strftime(p, 4 + 1, "%Y", tm))
607         goto fail;
608     return(buf);

610 fail:
611     free(buf);
612     return(NULL);
613 }

615 char *
616 mandoc_normdate(struct mparse *parse, char *in, int ln, int pos)
617 {
618     char           *out;
619     time_t         t;

621     if (NULL == in || '\0' == *in ||
622         0 == strcmp(in, "$" "Mdocdate$")) {
623         mandoc_msg(MANDOCERR_NODATE, parse, ln, pos, NULL);
624         time(&t);
625     }
626     else if (a2time(&t, "%Y-%m-%d", in))
627         t = 0;
628     else if (!a2time(&t, "$" "Mdocdate: %b %d %Y $" , in) &&
629             !a2time(&t, "%b %d, %Y", in)) {
630         mandoc_msg(MANDOCERR_BADDATE, parse, ln, pos, NULL);
631         t = 0;
632     }
633     out = t ? time2a(t) : NULL;
634     return(out ? out : mandoc_strdup(in));
635 }

637 int
638 mandoc_eos(const char *p, size_t sz, int enclosed)
639 {
640     const char *q;
641     int found;

643     if (0 == sz)
644         return(0);

646     /*
647     * End-of-sentence recognition must include situations where
648     * some symbols, such as '\)', allow prior EOS punctuation to
649     * propagate outward.
650     */

```

```

652     found = 0;
653     for (q = p + (int)sz - 1; q >= p; q--) {
654         switch (*q) {
655             case ('\0'):
656                 /* FALLTHROUGH */
657             case ('\'''):
658                 /* FALLTHROUGH */
659             case (']'):
660                 /* FALLTHROUGH */
661             case (')'):
662                 if (0 == found)
663                     enclosed = 1;
664                 break;
665             case ('.'):
666                 /* FALLTHROUGH */
667             case ('!'):
668                 /* FALLTHROUGH */
669             case ('?'):
670                 found = 1;
671                 break;
672             default:
673                 return(found && !enclosed || isalnum((unsigned char)*q)
674                )
675         }
676     }
677     return(found && !enclosed);
678 }
679
680 /*
681  * Find out whether a line is a macro line or not.  If it is, adjust the
682  * current position and return one; if it isn't, return zero and don't
683  * change the current position.
684  */
685 int
686 mandoc_getcontrol(const char *cp, int *ppos)
687 {
688     int         pos;
689
690     pos = *ppos;
691
692     if ('\0' == cp[pos] && '.' == cp[pos + 1])
693         pos += 2;
694     else if ('.' == cp[pos] || '\'' == cp[pos])
695         pos++;
696     else
697         return(0);
698
699     while (' ' == cp[pos] || '\t' == cp[pos])
700         pos++;
701
702     *ppos = pos;
703     return(1);
704 }
705
706 /*
707  * Convert a string to a long that may not be <0.
708  * If the string is invalid, or is less than 0, return -1.
709  */
710 int
711 mandoc_strntoi(const char *p, size_t sz, int base)
712 {
713     char        buf[32];
714     char        *ep;
715     long        v;

```

```

717     if (sz > 31)
718         return(-1);
719
720     memcpy(buf, p, sz);
721     buf[(int)sz] = '\0';
722
723     errno = 0;
724     v = strtol(buf, &ep, base);
725
726     if (buf[0] == '\0' || *ep != '\0')
727         return(-1);
728
729     if (v > INT_MAX)
730         v = INT_MAX;
731     if (v < INT_MIN)
732         v = INT_MIN;
733
734     return((int)v);
735 }

```



```

*****
13893 Sat Jul 19 19:24:30 2014
new/usr/src/cmd/mandoc/mandoc.h
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: mandoc.h,v 1.99 2012/02/16 20:51:31 joerg Exp $ */
2 /*
3  * Copyright (c) 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef MANDOC_H
18 #define MANDOC_H
19
20 #define ASCII_NBRSP      31 /* non-breaking space */
21 #define ASCII_HYPH      30 /* breakable hyphen */
22
23 /*
24  * Status level. This refers to both internal status (i.e., whilst
25  * running, when warnings/errors are reported) and an indicator of a
26  * threshold of when to halt (when said internal state exceeds the
27  * threshold).
28  */
29 enum mandoclevel {
30     MANDOCLEVEL_OK = 0,
31     MANDOCLEVEL_RESERVED,
32     MANDOCLEVEL_WARNING, /* warnings: syntax, whitespace, etc. */
33     MANDOCLEVEL_ERROR, /* input has been thrown away */
34     MANDOCLEVEL_FATAL, /* input is borked */
35     MANDOCLEVEL_BADARG, /* bad argument in invocation */
36     MANDOCLEVEL_SYSERR, /* system error */
37     MANDOCLEVEL_MAX
38 };
39
40 /*
41  * All possible things that can go wrong within a parse, be it libroff,
42  * libmdoc, or libman.
43  */
44 enum mandocerr {
45     MANDOCERR_OK,
46
47     MANDOCERR_WARNING, /* ===== start of warnings ===== */
48
49     /* related to the prologue */
50     MANDOCERR_NOTITLE, /* no title in document */
51     MANDOCERR_UPPERCASE, /* document title should be all caps */
52     MANDOCERR_BADMSEC, /* unknown manual section */
53     MANDOCERR_NODATE, /* date missing, using today's date */
54     MANDOCERR_BADDATE, /* cannot parse date, using it verbatim */
55     MANDOCERR_PROLOGOOO, /* prologue macros out of order */
56     MANDOCERR_PROLOGREP, /* duplicate prologue macro */

```

```

57 MANDOCERR_BADPROLOG, /* macro not allowed in prologue */
58 MANDOCERR_BADBODY, /* macro not allowed in body */
59
60 /* related to document structure */
61 MANDOCERR_SO, /* .so is fragile, better use ln(1) */
62 MANDOCERR_NAMESECFIRST, /* NAME section must come first */
63 MANDOCERR_BADNAMESEC, /* bad NAME section contents */
64 MANDOCERR_NONAME, /* manual name not yet set */
65 MANDOCERR_SECOOO, /* sections out of conventional order */
66 MANDOCERR_SECREP, /* duplicate section name */
67 MANDOCERR_SECMSEC, /* section not in conventional manual section */
68
69 /* related to macros and nesting */
70 MANDOCERR_MACROOBS, /* skipping obsolete macro */
71 MANDOCERR_IGNPAR, /* skipping paragraph macro */
72 MANDOCERR_IGNNS, /* skipping no-space macro */
73 MANDOCERR_SCOPENEST, /* blocks badly nested */
74 MANDOCERR_CHILD, /* child violates parent syntax */
75 MANDOCERR_NESTEDDISP, /* nested displays are not portable */
76 MANDOCERR_SCOPEREPR, /* already in literal mode */
77 MANDOCERR_LINESCOPE, /* line scope broken */
78
79 /* related to missing macro arguments */
80 MANDOCERR_MACROEMPTY, /* skipping empty macro */
81 MANDOCERR_ARGCWRN, /* argument count wrong */
82 MANDOCERR_DISPTYPE, /* missing display type */
83 MANDOCERR_LISTFIRST, /* list type must come first */
84 MANDOCERR_NOWIDTHARG, /* tag lists require a width argument */
85 MANDOCERR_FONTTYPE, /* missing font type */
86 MANDOCERR_WNOSCOPE, /* skipping end of block that is not open */
87
88 /* related to bad macro arguments */
89 MANDOCERR_IGNARGV, /* skipping argument */
90 MANDOCERR_ARGVREP, /* duplicate argument */
91 MANDOCERR_DISPREP, /* duplicate display type */
92 MANDOCERR_LISTREP, /* duplicate list type */
93 MANDOCERR_BADATT, /* unknown AT&T UNIX version */
94 MANDOCERR_BADBOOL, /* bad Boolean value */
95 MANDOCERR_BADFONT, /* unknown font */
96 MANDOCERR_BADSTANDARD, /* unknown standard specifier */
97 MANDOCERR_BADWIDTH, /* bad width argument */
98
99 /* related to plain text */
100 MANDOCERR_NOBLANKLN, /* blank line in non-literal context */
101 MANDOCERR_BADTAB, /* tab in non-literal context */
102 MANDOCERR_EOLNSPACE, /* end of line whitespace */
103 MANDOCERR_BADCOMMENT, /* bad comment style */
104 MANDOCERR_BADESCAPE, /* unknown escape sequence */
105 MANDOCERR_BADQUOTE, /* unterminated quoted string */
106
107 /* related to equations */
108 MANDOCERR_EQNQUOTE, /* unexpected literal in equation */
109
110 MANDOCERR_ERROR, /* ===== start of errors ===== */
111
112 /* related to equations */
113 MANDOCERR_EQNNSCOPE, /* unexpected equation scope closure */
114 MANDOCERR_EQNSCOPE, /* equation scope open on exit */
115 MANDOCERR_EQNBADSCOPE, /* overlapping equation scopes */
116 MANDOCERR_EQNEOF, /* unexpected end of equation */
117 MANDOCERR_EQNSYNT, /* equation syntax error */
118
119 /* related to tables */
120 MANDOCERR_TBL, /* bad table syntax */
121 MANDOCERR_TBLOPT, /* bad table option */
122 MANDOCERR_TBLAYOUT, /* bad table layout */

```

```

123 MANDOCERR_TBLNOLAYOUT, /* no table layout cells specified */
124 MANDOCERR_TBLNODATA, /* no table data cells specified */
125 MANDOCERR_TBLIGNDATA, /* ignore data in cell */
126 MANDOCERR_TBLEBLOCK, /* data block still open */
127 MANDOCERR_TBLEXTRADAT, /* ignoring extra data cells */

129 MANDOCERR_ROFFLOOP, /* input stack limit exceeded, infinite loop? */
130 MANDOCERR_BADCHAR, /* skipping bad character */
131 MANDOCERR_NAMESC, /* escaped character not allowed in a name */
132 MANDOCERR_NOTEXT, /* skipping text before the first section header */
133 MANDOCERR_MACRO, /* skipping unknown macro */
134 MANDOCERR_REQUEST, /* NOT IMPLEMENTED: skipping request */
135 MANDOCERR_ARGCOUNT, /* argument count wrong */
136 MANDOCERR_NOSCOPE, /* skipping end of block that is not open */
137 MANDOCERR_SCOPEBROKEN, /* missing end of block */
138 MANDOCERR_SCOPEEXIT, /* scope open on exit */
139 MANDOCERR_UNAME, /* uname(3) system call failed */
140 /* FIXME: merge following with MANDOCERR_ARGCOUNT */
141 MANDOCERR_NOARGS, /* macro requires line argument(s) */
142 MANDOCERR_NOBODY, /* macro requires body argument(s) */
143 MANDOCERR_NOARGV, /* macro requires argument(s) */
144 MANDOCERR_LISTTYPE, /* missing list type */
145 MANDOCERR_ARGSLOST, /* line argument(s) will be lost */
146 MANDOCERR_BODYLOST, /* body argument(s) will be lost */

148 MANDOCERR_FATAL, /* ===== start of fatal errors ===== */

150 MANDOCERR_NOTMANUAL, /* manual isn't really a manual */
151 MANDOCERR_COLUMNS, /* column syntax is inconsistent */
152 MANDOCERR_BADDISP, /* NOT IMPLEMENTED: .Bd -file */
153 MANDOCERR_SYNTARGVCOUNT, /* argument count wrong, violates syntax */
154 MANDOCERR_SYNTCHILD, /* child violates parent syntax */
155 MANDOCERR_SYNTARGCOUNT, /* argument count wrong, violates syntax */
156 MANDOCERR_SOPATH, /* NOT IMPLEMENTED: .so with absolute path or "." */
157 MANDOCERR_NODOCBODY, /* no document body */
158 MANDOCERR_NODOCPROLOG, /* no document prologue */
159 MANDOCERR_MEM, /* static buffer exhausted */
160 MANDOCERR_MAX
161 };

163 struct tbl {
164     char          tab; /* cell-separator */
165     char          decimal; /* decimal point */
166     int           linesize;
167     int           opts;
168 #define TBL_OPT_CENTRE (1 << 0)
169 #define TBL_OPT_EXPAND (1 << 1)
170 #define TBL_OPT_BOX (1 << 2)
171 #define TBL_OPT_DBOX (1 << 3)
172 #define TBL_OPT_ALLBOX (1 << 4)
173 #define TBL_OPT_NOKEEP (1 << 5)
174 #define TBL_OPT_NOSPACE (1 << 6)
175     int           cols; /* number of columns */
176 };

178 enum tbl_headt {
179     TBL_HEAD_DATA, /* plug in data from tbl_dat */
180     TBL_HEAD_VERT, /* vertical spacer */
181     TBL_HEAD_DVERT /* double-vertical spacer */
182 };

184 /*
185 * The head of a table specifies all of its columns. When formatting a
186 * tbl_span, iterate over these and plug in data from the tbl_span when
187 * appropriate, using tbl_cell as a guide to placement.
188 */

```

```

189 struct tbl_head {
190     enum tbl_headt    pos;
191     int               id; /* 0 <= unique id < cols */
192     struct tbl_head  *next;
193     struct tbl_head  *prev;
194 };

196 enum tbl_cellt {
197     TBL_CELL_CENTRE, /* c, C */
198     TBL_CELL_RIGHT, /* r, R */
199     TBL_CELL_LEFT, /* l, L */
200     TBL_CELL_NUMBER, /* n, N */
201     TBL_CELL_SPAN, /* s, S */
202     TBL_CELL_LONG, /* a, A */
203     TBL_CELL_DOWN, /* ^ */
204     TBL_CELL_HORIZ, /* -, - */
205     TBL_CELL_DHORIZ, /* = */
206     TBL_CELL_VERT, /* | */
207     TBL_CELL_DVERT, /* || */
208     TBL_CELL_MAX
209 };

211 /*
212 * A cell in a layout row.
213 */
214 struct tbl_cell {
215     struct tbl_cell *next;
216     enum tbl_cellt  pos;
217     size_t          spacing;
218     int             flags;
219 #define TBL_CELL_TALIGN (1 << 0) /* t, T */
220 #define TBL_CELL_BALIGN (1 << 1) /* d, D */
221 #define TBL_CELL_BOLD (1 << 2) /* fB, B, b */
222 #define TBL_CELL_ITALIC (1 << 3) /* fI, I, i */
223 #define TBL_CELL_EQUAL (1 << 4) /* e, E */
224 #define TBL_CELL_UP (1 << 5) /* u, U */
225 #define TBL_CELL_WIGN (1 << 6) /* z, Z */
226     struct tbl_head *head;
227 };

229 /*
230 * A layout row.
231 */
232 struct tbl_row {
233     struct tbl_row *next;
234     struct tbl_cell *first;
235     struct tbl_cell *last;
236 };

238 enum tbl_datt {
239     TBL_DATA_NONE, /* has no data */
240     TBL_DATA_DATA, /* consists of data/string */
241     TBL_DATA_HORIZ, /* horizontal line */
242     TBL_DATA_DHORIZ, /* double-horizontal line */
243     TBL_DATA_NHORIZ, /* squeezed horizontal line */
244     TBL_DATA_NDHORIZ /* squeezed double-horizontal line */
245 };

247 /*
248 * A cell within a row of data. The "string" field contains the actual
249 * string value that's in the cell. The rest is layout.
250 */
251 struct tbl_dat {
252     struct tbl_cell *layout; /* layout cell */
253     int             spans; /* how many spans follow */
254     struct tbl_dat  *next;

```

```

255     char          *string; /* data (NULL if not TBL_DATA_DATA) */
256     enum tbl_datt  pos;
257 };

259 enum   tbl_spant {
260     TBL_SPAN_DATA, /* span consists of data */
261     TBL_SPAN_HORIZ, /* span is horizontal line */
262     TBL_SPAN_DHORIZ /* span is double horizontal line */
263 };

265 /*
266  * A row of data in a table.
267  */
268 struct tbl_span {
269     struct tbl      *tbl;
270     struct tbl_head *head;
271     struct tbl_row  *layout; /* layout row */
272     struct tbl_dat  *first;
273     struct tbl_dat  *last;
274     int             line; /* parse line */
275     int             flags;
276 #define TBL_SPAN_FIRST (1 << 0)
277 #define TBL_SPAN_LAST (1 << 1)
278     enum tbl_spant  pos;
279     struct tbl_span *next;
280 };

282 enum   eqn_boxst {
283     EQN_ROOT, /* root of parse tree */
284     EQN_TEXT, /* text (number, variable, whatever) */
285     EQN_SUBEXPR, /* nested 'eqn' subexpression */
286     EQN_LIST, /* subexpressions list */
287     EQN_MATRIX /* matrix subexpression */
288 };

290 enum   eqn_markt {
291     EQNMARK_NONE = 0,
292     EQNMARK_DOT,
293     EQNMARK_DOTDOT,
294     EQNMARK_HAT,
295     EQNMARK_TILDE,
296     EQNMARK_VEC,
297     EQNMARK_DYAD,
298     EQNMARK_BAR,
299     EQNMARK_UNDER,
300     EQNMARK_MAX
301 };

303 enum   eqn_fontt {
304     EQNFONT_NONE = 0,
305     EQNFONT_ROMAN,
306     EQNFONT_BOLD,
307     EQNFONT_FAT,
308     EQNFONT_ITALIC,
309     EQNFONT_MAX
310 };

312 enum   eqn_post {
313     EQNPOS_NONE = 0,
314     EQNPOS_OVER,
315     EQNPOS_SUP,
316     EQNPOS_SUB,
317     EQNPOS_TO,
318     EQNPOS_FROM,
319     EQNPOS_MAX
320 };

```

```

322 enum   eqn_pilet {
323     EQNPILE_NONE = 0,
324     EQNPILE_PILE,
325     EQNPILE_CPILE,
326     EQNPILE_RPILE,
327     EQNPILE_LPILE,
328     EQNPILE_COL,
329     EQNPILE_CCOL,
330     EQNPILE_RCOL,
331     EQNPILE_LCOL,
332     EQNPILE_MAX
333 };

335 /*
336  * A "box" is a parsed mathematical expression as defined by the eqn.7
337  * grammar.
338  */
339 struct eqn_box {
340     int             size; /* font size of expression */
341 #define EQN_DEFSIZE INT_MIN
342     enum eqn_boxst  type; /* type of node */
343     struct eqn_box  *first; /* first child node */
344     struct eqn_box  *last; /* last child node */
345     struct eqn_box  *next; /* node sibling */
346     struct eqn_box  *parent; /* node sibling */
347     char            *text; /* text (or NULL) */
348     char            *left;
349     char            *right;
350     enum eqn_post   pos; /* position of next box */
351     enum eqn_markt  mark; /* a mark about the box */
352     enum eqn_fontt  font; /* font of box */
353     enum eqn_pilet  pile; /* equation piling */
354 };

356 /*
357  * An equation consists of a tree of expressions starting at a given
358  * line and position.
359  */
360 struct eqn {
361     char            *name; /* identifier (or NULL) */
362     struct eqn_box  *root; /* root mathematical expression */
363     int             ln; /* invocation line */
364     int             pos; /* invocation position */
365 };

367 /*
368  * The type of parse sequence. This value is usually passed via the
369  * mandoc(1) command line of -man and -mdoc. It's almost exclusively
370  * -mandoc but the others have been retained for compatibility.
371  */
372 enum   mparset {
373     MPARSE_AUTO, /* magically determine the document type */
374     MPARSE_MDOC, /* assume -mdoc */
375     MPARSE_MAN /* assume -man */
376 };

378 enum   mandoc_esc {
379     ESCAPE_ERROR = 0, /* bail! unparsable escape */
380     ESCAPE_IGNORE, /* escape to be ignored */
381     ESCAPE_SPECIAL, /* a regular special character */
382     ESCAPE_FONT, /* a generic font mode */
383     ESCAPE_FONTBOLD, /* bold font mode */
384     ESCAPE_FONTITALIC, /* italic font mode */
385     ESCAPE_FONTROMAN, /* roman font mode */
386     ESCAPE_FONTPREV, /* previous font mode */

```

```
387     ESCAPE_NUMBERED, /* a numbered glyph */
388     ESCAPE_UNICODE, /* a unicode codepoint */
389     ESCAPE_NOSPACE /* suppress space if the last on a line */
390 };

392 typedef void      (*mandocmsg)(enum mandocerr, enum mandoclevel,
393                               const char *, int, int, const char *);

395 struct  mparse;
396 struct  mchars;
397 struct  mdoc;
398 struct  man;

400 __BEGIN_DECLS

402 void      *mandoc_calloc(size_t, size_t);
403 enum mandoc_esc  mandoc_escape(const char **, const char **, int *);
404 void      *mandoc_malloc(size_t);
405 void      *mandoc_realloc(void *, size_t);
406 char      *mandoc_strdup(const char *);
407 char      *mandoc_strndup(const char *, size_t);
408 struct mchars  *mchars_alloc(void);
409 void      mchars_free(struct mchars *);
410 char      mchars_num2char(const char *, size_t);
411 int       mchars_num2uc(const char *, size_t);
412 int       mchars_spec2cp(const struct mchars *,
413                          const char *, size_t);
414 const char *mchars_spec2str(const struct mchars *,
415                             const char *, size_t, size_t *);
416 struct mparse  *mparse_alloc(enum mparset,
417                              enum mandoclevel, mandocmsg, void *);
418 void      mparse_free(struct mparse *);
419 void      mparse_keep(struct mparse *);
420 enum mandoclevel  mparse_readfd(struct mparse *, int, const char *);
421 enum mandoclevel  mparse_readmem(struct mparse *, const void *, size_t,
422                                  const char *);
423 void      mparse_reset(struct mparse *);
424 void      mparse_result(struct mparse *,
425                          struct mdoc **, struct man **);
426 const char *mparse_getkeep(const struct mparse *);
427 const char *mparse_strerror(enum mandocerr);
428 const char *mparse_strlevel(enum mandoclevel);

430 __END_DECLS

432 #endif /*!MANDOC_H*/
```

```

*****
20580 Sat Jul 19 19:24:30 2014
new/usr/src/cmd/mandoc/mdoc.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: mdoc.c,v 1.196 2011/09/30 00:13:28 schwarze Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2010 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <sys/types.h>
23
24 #include <assert.h>
25 #include <stdarg.h>
26 #include <stdio.h>
27 #include <stdlib.h>
28 #include <string.h>
29 #include <time.h>
30
31 #include "mdoc.h"
32 #include "mandoc.h"
33 #include "libmdoc.h"
34 #include "libmandoc.h"
35
36 const char *const __mdoc_macronames[MDOC_MAX] = {
37     "Ap",      "Dd",      "Dt",      "Os",
38     "Sh",      "Ss",      "Pp",      "Dl",
39     "Dl",      "Bd",      "Ed",      "Bl",
40     "El",      "It",      "Ad",      "An",
41     "Ar",      "Cd",      "Cm",      "Dv",
42     "Er",      "Ev",      "Ex",      "Fa",
43     "Fd",      "Fl",      "Fn",      "Ft",
44     "Ic",      "In",      "Li",      "Nd",
45     "Nm",      "Op",      "Pa",      "Pa",
46     "Rv",      "St",      "Va",      "Vt",
47     /* LINTED */
48     "Xr",      "%A",      "%B",      "%D",
49     /* LINTED */
50     "%I",      "%J",      "%N",      "%O",
51     /* LINTED */
52     "%P",      "%R",      "%T",      "%V",
53     "Ac",      "Ac",      "Aq",      "At",
54     "Bc",      "Bf",      "Bo",      "Bq",
55     "Bsx",     "Bx",      "Db",      "Dc",
56     "Do",      "Dq",      "Ec",      "Ef",

```

```

57     "Em",      "Eo",      "Fx",      "Ms",
58     "No",      "Ns",      "Nx",      "Ox",
59     "Pc",      "Pf",      "Po",      "Pq",
60     "Qc",      "Ql",      "Qo",      "Qq",
61     "Re",      "Rs",      "Sc",      "So",
62     "Sq",      "Sm",      "Sx",      "Sy",
63     "Tn",      "Ux",      "Xc",      "Xo",
64     "Fc",      "Fc",      "Oo",      "Oc",
65     "Bk",      "Ek",      "Bt",      "Hf",
66     "Fr",      "Ud",      "Lb",      "Lp",
67     "Lk",      "Mt",      "Brq",     "Bro",
68     /* LINTED */
69     "Brc",      "%C",      "Es",      "En",
70     /* LINTED */
71     "Dx",      "%Q",      "br",      "sp",
72     /* LINTED */
73     "%U",      "Ta"
74     };
75
76 const char *const __mdoc_argnames[MDOC_ARG_MAX] = {
77     "split",     "nosplit",
78     "unfilled", "literal",
79     "offset",    "bullet",
80     "hyphen",    "item",
81     "tag",       "diag",
82     "ohang",    "inset",
83     "width",    "compact",
84     "filled",   "words",
85     "symbolic", "nested",
86     "centered"
87     };
88
89 const char *const *mdoc_macronames = __mdoc_macronames;
90 const char *const *mdoc_argnames = __mdoc_argnames;
91
92 static void mdoc_node_free(struct mdoc_node *);
93 static void mdoc_node_unlink(struct mdoc *,
94     struct mdoc_node *);
95 static void mdoc_freel(struct mdoc_node *);
96 static void mdoc_alloc1(struct mdoc *);
97 static struct mdoc_node *mdoc_alloc(struct mdoc *, int, int,
98     enum mdoct, enum mdoc_type);
99 static int node_append(struct mdoc *,
100     struct mdoc_node *);
101 #if 0
102 static int mdoc_preptext(struct mdoc *, int, char *, int);
103 #endif
104 static int mdoc_ptext(struct mdoc *, int, char *, int);
105 static int mdoc_pmacro(struct mdoc *, int, char *, int);
106
107 const struct mdoc_node *
108 mdoc_node(const struct mdoc *m)
109 {
110     assert( ! (MDOC_HALT & m->flags));
111     return(m->first);
112 }
113
114 const struct mdoc_meta *
115 mdoc_meta(const struct mdoc *m)
116 {
117     assert( ! (MDOC_HALT & m->flags));
118     return(&m->meta);
119 }

```

```

124 /*
125  * Frees volatile resources (parse tree, meta-data, fields).
126  */
127 static void
128 mdoc_freel(struct mdoc *mdoc)
129 {
131     if (mdoc->first)
132         mdoc_node_delete(mdoc, mdoc->first);
133     if (mdoc->meta.title)
134         free(mdoc->meta.title);
135     if (mdoc->meta.os)
136         free(mdoc->meta.os);
137     if (mdoc->meta.name)
138         free(mdoc->meta.name);
139     if (mdoc->meta.arch)
140         free(mdoc->meta.arch);
141     if (mdoc->meta.vol)
142         free(mdoc->meta.vol);
143     if (mdoc->meta.msec)
144         free(mdoc->meta.msec);
145     if (mdoc->meta.date)
146         free(mdoc->meta.date);
147 }

150 /*
151  * Allocate all volatile resources (parse tree, meta-data, fields).
152  */
153 static void
154 mdoc_alloc1(struct mdoc *mdoc)
155 {
157     memset(&mdoc->meta, 0, sizeof(struct mdoc_meta));
158     mdoc->flags = 0;
159     mdoc->lastnamed = mdoc->lastsec = SEC_NONE;
160     mdoc->last = mandoc_calloc(1, sizeof(struct mdoc_node));
161     mdoc->first = mdoc->last;
162     mdoc->last->type = MDOC_ROOT;
163     mdoc->last->tok = MDOC_MAX;
164     mdoc->next = MDOC_NEXT_CHILD;
165 }

168 /*
169  * Free up volatile resources (see mdoc_freel()) then re-initialises the
170  * data with mdoc_alloc1(). After invocation, parse data has been reset
171  * and the parser is ready for re-invocation on a new tree; however,
172  * cross-parse non-volatile data is kept intact.
173  */
174 void
175 mdoc_reset(struct mdoc *mdoc)
176 {
178     mdoc_freel(mdoc);
179     mdoc_alloc1(mdoc);
180 }

183 /*
184  * Completely free up all volatile and non-volatile parse resources.
185  * After invocation, the pointer is no longer usable.
186  */
187 void
188 mdoc_free(struct mdoc *mdoc)

```

```

189 {
191     mdoc_freel(mdoc);
192     free(mdoc);
193 }

196 /*
197  * Allocate volatile and non-volatile parse resources.
198  */
199 struct mdoc *
200 mdoc_alloc(struct roff *roff, struct mparse *parse)
201 {
202     struct mdoc *p;
204     p = mandoc_calloc(1, sizeof(struct mdoc));
206     p->parse = parse;
207     p->roff = roff;
209     mdoc_hash_init();
210     mdoc_alloc1(p);
211     return(p);
212 }

215 /*
216  * Climb back up the parse tree, validating open scopes. Mostly calls
217  * through to macro_end() in macro.c.
218  */
219 int
220 mdoc_endparse(struct mdoc *m)
221 {
223     assert( ! (MDOC_HALT & m->flags));
224     if (mdoc_macroend(m))
225         return(1);
226     m->flags |= MDOC_HALT;
227     return(0);
228 }

230 int
231 mdoc_addeqn(struct mdoc *m, const struct eqn *ep)
232 {
233     struct mdoc_node *n;
235     assert( ! (MDOC_HALT & m->flags));
237     /* No text before an initial macro. */
239     if (SEC_NONE == m->lastnamed) {
240         mdoc_pmsg(m, ep->ln, ep->pos, MANDOCERR_NOTEXT);
241         return(1);
242     }
244     n = node_alloc(m, ep->ln, ep->pos, MDOC_MAX, MDOC_EQN);
245     n->eqn = ep;
247     if ( ! node_append(m, n))
248         return(0);
250     m->next = MDOC_NEXT_SIBLING;
251     return(1);
252 }

254 int

```

```

255 mdoc_addspan(struct mdoc *m, const struct tbl_span *sp)
256 {
257     struct mdoc_node *n;
258
259     assert( ! (MDOC_HALT & m->flags));
260
261     /* No text before an initial macro. */
262
263     if (SEC_NONE == m->lastnamed) {
264         mdoc_pmsg(m, sp->line, 0, MANDOCERR_NOTEXT);
265         return(1);
266     }
267
268     n = node_alloc(m, sp->line, 0, MDOC_MAX, MDOC_TBL);
269     n->span = sp;
270
271     if ( ! node_append(m, n))
272         return(0);
273
274     m->next = MDOC_NEXT_SIBLING;
275     return(1);
276 }
277
278
279 /*
280 * Main parse routine. Parses a single line -- really just hands off to
281 * the macro (mdoc_pmacro()) or text parser (mdoc_ptext()).
282 */
283 int
284 mdoc_parsein(struct mdoc *m, int ln, char *buf, int offs)
285 {
286
287     assert( ! (MDOC_HALT & m->flags));
288
289     m->flags |= MDOC_NEWLINE;
290
291     /*
292     * Let the roff nS register switch SYNOPSIS mode early,
293     * such that the parser knows at all times
294     * whether this mode is on or off.
295     * Note that this mode is also switched by the Sh macro.
296     */
297     if (roff_regisset(m->roff, REG_nS)) {
298         if (roff_regget(m->roff, REG_nS))
299             m->flags |= MDOC_SYNOPSIS;
300         else
301             m->flags &= ~MDOC_SYNOPSIS;
302     }
303
304     return(mandoc_getcontrol(buf, &offs) ?
305         mdoc_pmacro(m, ln, buf, offs) :
306         mdoc_ptext(m, ln, buf, offs));
307 }
308
309 int
310 mdoc_macro(MACRO_PROT_ARGS)
311 {
312     assert(tok < MDOC_MAX);
313
314     /* If we're in the body, deny prologue calls. */
315
316     if (MDOC_PROLOGUE & mdoc_macros[tok].flags &&
317         MDOC_PBODY & m->flags) {
318         mdoc_pmsg(m, line, ppos, MANDOCERR_BADBODY);
319         return(1);
320     }

```

```

322     /* If we're in the prologue, deny "body" macros. */
323
324     if ( ! (MDOC_PROLOGUE & mdoc_macros[tok].flags) &&
325         ! (MDOC_PBODY & m->flags)) {
326         mdoc_pmsg(m, line, ppos, MANDOCERR_BADPROLOG);
327         if (NULL == m->meta.msec)
328             m->meta.msec = mandoc_strdup("1");
329         if (NULL == m->meta.title)
330             m->meta.title = mandoc_strdup("UNKNOWN");
331         if (NULL == m->meta.vol)
332             m->meta.vol = mandoc_strdup("LOCAL");
333         if (NULL == m->meta.os)
334             m->meta.os = mandoc_strdup("LOCAL");
335         if (NULL == m->meta.date)
336             m->meta.date = mandoc_normdate
337                 (m->parse, NULL, line, ppos);
338         m->flags |= MDOC_PBODY;
339     }
340
341     return((*mdoc_macros[tok].fp)(m, tok, line, ppos, pos, buf));
342 }
343
344
345 static int
346 node_append(struct mdoc *mdoc, struct mdoc_node *p)
347 {
348
349     assert(mdoc->last);
350     assert(mdoc->first);
351     assert(MDOC_ROOT != p->type);
352
353     switch (mdoc->next) {
354     case (MDOC_NEXT_SIBLING):
355         mdoc->last->next = p;
356         p->prev = mdoc->last;
357         p->parent = mdoc->last->parent;
358         break;
359     case (MDOC_NEXT_CHILD):
360         mdoc->last->child = p;
361         p->parent = mdoc->last;
362         break;
363     default:
364         abort();
365         /* NOTREACHED */
366     }
367
368     p->parent->nchild++;
369
370     /*
371     * Copy over the normalised-data pointer of our parent. Not
372     * everybody has one, but copying a null pointer is fine.
373     */
374
375     switch (p->type) {
376     case (MDOC_BODY):
377         /* FALLTHROUGH */
378     case (MDOC_TAIL):
379         /* FALLTHROUGH */
380     case (MDOC_HEAD):
381         p->norm = p->parent->norm;
382         break;
383     default:
384         break;
385     }

```

```

387     if ( ! mdoc_valid_pre(mdoc, p))
388         return(0);

390     switch (p->type) {
391     case (MDOC_HEAD):
392         assert(MDOC_BLOCK == p->parent->type);
393         p->parent->head = p;
394         break;
395     case (MDOC_TAIL):
396         assert(MDOC_BLOCK == p->parent->type);
397         p->parent->tail = p;
398         break;
399     case (MDOC_BODY):
400         if (p->end)
401             break;
402         assert(MDOC_BLOCK == p->parent->type);
403         p->parent->body = p;
404         break;
405     default:
406         break;
407     }

409     mdoc->last = p;

411     switch (p->type) {
412     case (MDOC_TBL):
413         /* FALLTHROUGH */
414     case (MDOC_TEXT):
415         if ( ! mdoc_valid_post(mdoc))
416             return(0);
417         break;
418     default:
419         break;
420     }

422     return(1);
423 }

426 static struct mdoc_node *
427 node_alloc(struct mdoc *m, int line, int pos,
428            enum mdoct tok, enum mdoc_type type)
429 {
430     struct mdoc_node *p;

432     p = mdoc_calloc(1, sizeof(struct mdoc_node));
433     p->sec = m->lastsec;
434     p->line = line;
435     p->pos = pos;
436     p->tok = tok;
437     p->type = type;

439     /* Flag analysis. */

441     if (MDOC_SYNOPSIS & m->flags)
442         p->flags |= MDOC_SYNPRETTY;
443     else
444         p->flags &= ~MDOC_SYNPRETTY;
445     if (MDOC_NEWLINE & m->flags)
446         p->flags |= MDOC_LINE;
447     m->flags &= ~MDOC_NEWLINE;

449     return(p);
450 }

```

```

453 int
454 mdoc_tail_alloc(struct mdoc *m, int line, int pos, enum mdoct tok)
455 {
456     struct mdoc_node *p;

458     p = node_alloc(m, line, pos, tok, MDOC_TAIL);
459     if ( ! node_append(m, p))
460         return(0);
461     m->next = MDOC_NEXT_CHILD;
462     return(1);
463 }

466 int
467 mdoc_head_alloc(struct mdoc *m, int line, int pos, enum mdoct tok)
468 {
469     struct mdoc_node *p;

471     assert(m->first);
472     assert(m->last);

474     p = node_alloc(m, line, pos, tok, MDOC_HEAD);
475     if ( ! node_append(m, p))
476         return(0);
477     m->next = MDOC_NEXT_CHILD;
478     return(1);
479 }

482 int
483 mdoc_body_alloc(struct mdoc *m, int line, int pos, enum mdoct tok)
484 {
485     struct mdoc_node *p;

487     p = node_alloc(m, line, pos, tok, MDOC_BODY);
488     if ( ! node_append(m, p))
489         return(0);
490     m->next = MDOC_NEXT_CHILD;
491     return(1);
492 }

495 int
496 mdoc_endbody_alloc(struct mdoc *m, int line, int pos, enum mdoct tok,
497                   struct mdoc_node *body, enum mdoc_endbody end)
498 {
499     struct mdoc_node *p;

501     p = node_alloc(m, line, pos, tok, MDOC_BODY);
502     p->pending = body;
503     p->end = end;
504     if ( ! node_append(m, p))
505         return(0);
506     m->next = MDOC_NEXT_SIBLING;
507     return(1);
508 }

511 int
512 mdoc_block_alloc(struct mdoc *m, int line, int pos,
513                 enum mdoct tok, struct mdoc_arg *args)
514 {
515     struct mdoc_node *p;

517     p = node_alloc(m, line, pos, tok, MDOC_BLOCK);
518     p->args = args;

```



```

519     if (p->args)
520         (args->refcnt)++;

522     switch (tok) {
523     case (MDOC_Bd):
524         /* FALLTHROUGH */
525     case (MDOC_BF):
526         /* FALLTHROUGH */
527     case (MDOC_Bl):
528         /* FALLTHROUGH */
529     case (MDOC_Rs):
530         p->norm = mandoc_calloc(1, sizeof(union mdoc_data));
531         break;
532     default:
533         break;
534     }

536     if ( ! node_append(m, p))
537         return(0);
538     m->next = MDOC_NEXT_CHILD;
539     return(1);
540 }

543 int
544 mdoc_elem_alloc(struct mdoc *m, int line, int pos,
545                enum mdoctok tok, struct mdoc_arg *args)
546 {
547     struct mdoc_node *p;

549     p = node_alloc(m, line, pos, tok, MDOC_ELEM);
550     p->args = args;
551     if (p->args)
552         (args->refcnt)++;

554     switch (tok) {
555     case (MDOC_An):
556         p->norm = mandoc_calloc(1, sizeof(union mdoc_data));
557         break;
558     default:
559         break;
560     }

562     if ( ! node_append(m, p))
563         return(0);
564     m->next = MDOC_NEXT_CHILD;
565     return(1);
566 }

568 int
569 mdoc_word_alloc(struct mdoc *m, int line, int pos, const char *p)
570 {
571     struct mdoc_node *n;

573     n = node_alloc(m, line, pos, MDOC_MAX, MDOC_TEXT);
574     n->string = roff_strdup(m->roff, p);

576     if ( ! node_append(m, n))
577         return(0);

579     m->next = MDOC_NEXT_SIBLING;
580     return(1);
581 }

584 static void

```

```

585 mdoc_node_free(struct mdoc_node *p)
586 {
588     if (MDOC_BLOCK == p->type || MDOC_ELEM == p->type)
589         free(p->norm);
590     if (p->string)
591         free(p->string);
592     if (p->args)
593         mdoc_argv_free(p->args);
594     free(p);
595 }

598 static void
599 mdoc_node_unlink(struct mdoc *m, struct mdoc_node *n)
600 {
602     /* Adjust siblings. */

604     if (n->prev)
605         n->prev->next = n->next;
606     if (n->next)
607         n->next->prev = n->prev;

609     /* Adjust parent. */

611     if (n->parent) {
612         n->parent->nchild--;
613         if (n->parent->child == n)
614             n->parent->child = n->prev ? n->prev : n->next;
615         if (n->parent->last == n)
616             n->parent->last = n->prev ? n->prev : NULL;
617     }

619     /* Adjust parse point, if applicable. */

621     if (m && m->last == n) {
622         if (n->prev) {
623             m->last = n->prev;
624             m->next = MDOC_NEXT_SIBLING;
625         } else {
626             m->last = n->parent;
627             m->next = MDOC_NEXT_CHILD;
628         }
629     }

631     if (m && m->first == n)
632         m->first = NULL;
633 }

636 void
637 mdoc_node_delete(struct mdoc *m, struct mdoc_node *p)
638 {
640     while (p->child) {
641         assert(p->nchild);
642         mdoc_node_delete(m, p->child);
643     }
644     assert(0 == p->nchild);

646     mdoc_node_unlink(m, p);
647     mdoc_node_free(p);
648 }

650 #if 0

```

```

651 /*
652  * Pre-treat a text line.
653  * Text lines can consist of equations, which must be handled apart from
654  * the regular text.
655  * Thus, use this function to step through a line checking if it has any
656  * equations embedded in it.
657  * This must handle multiple equations AND equations that do not end at
658  * the end-of-line, i.e., will re-enter in the next roff parse.
659  */
660 static int
661 mdoc_pretext(struct mdoc *m, int line, char *buf, int offs)
662 {
663     char          *start, *end;
664     char          delim;
665
666     while ('\\0' != buf[offs]) {
667         /* Mark starting position if eqn is set. */
668         start = NULL;
669         if ('\\0' != (delim = roff_eqndelim(m->roff)))
670             if (NULL != (start = strchr(buf + offs, delim)))
671                 *start++ = '\\0';
672
673         /* Parse text as normal. */
674         if (!mdoc_ptext(m, line, buf, offs))
675             return(0);
676
677         /* Continue only if an equation exists. */
678         if (NULL == start)
679             break;
680
681         /* Read past the end of the equation. */
682         offs += start - (buf + offs);
683         assert(start == &buf[offs]);
684         if (NULL != (end = strchr(buf + offs, delim))) {
685             *end++ = '\\0';
686             while (' ' == *end)
687                 end++;
688         }
689
690         /* Parse the equation itself. */
691         roff_openeqn(m->roff, NULL, line, offs, buf);
692
693         /* Process a finished equation? */
694         if (roff_closeeqn(m->roff))
695             if (!mdoc_addeqn(m, roff_eqn(m->roff)))
696                 return(0);
697         offs += (end - (buf + offs));
698     }
699
700     return(1);
701 }
702 #endif
703
704 /*
705  * Parse free-form text, that is, a line that does not begin with the
706  * control character.
707  */
708 static int
709 mdoc_ptext(struct mdoc *m, int line, char *buf, int offs)
710 {
711     char          *c, *ws, *end;
712     struct mdoc_node *n;
713
714     /* No text before an initial macro. */
715
716     if (SEC_NONE == m->lastnamed) {

```

```

717         mdoc_pmsg(m, line, offs, MANDOCERR_NOTEXT);
718         return(1);
719     }
720
721     assert(m->last);
722     n = m->last;
723
724     /*
725     * Divert directly to list processing if we're encountering a
726     * columnar MDOC_BLOCK with or without a prior MDOC_BLOCK entry
727     * (a MDOC_BODY means it's already open, in which case we should
728     * process within its context in the normal way).
729     */
730
731     if (MDOC_Bl == n->tok && MDOC_BODY == n->type &&
732         LIST_column == n->norm->Bl.type) {
733         /* 'Bl' is open without any children. */
734         m->flags |= MDOC_FREECOL;
735         return(mdoc_macro(m, MDOC_It, line, offs, &offs, buf));
736     }
737
738     if (MDOC_It == n->tok && MDOC_BLOCK == n->type &&
739         NULL != n->parent &&
740         MDOC_Bl == n->parent->tok &&
741         LIST_column == n->parent->Bl.type) {
742         /* 'Bl' has block-level 'It' children. */
743         m->flags |= MDOC_FREECOL;
744         return(mdoc_macro(m, MDOC_It, line, offs, &offs, buf));
745     }
746
747     /*
748     * Search for the beginning of unescaped trailing whitespace (ws)
749     * and for the first character not to be output (end).
750     */
751
752     /* FIXME: replace with strcspn(). */
753     ws = NULL;
754     for (c = end = buf + offs; *c; c++) {
755         switch (*c) {
756             case ' ':
757                 if (NULL == ws)
758                     ws = c;
759                 continue;
760             case '\\t':
761                 /*
762                 * Always warn about trailing tabs,
763                 * even outside literal context,
764                 * where they should be put on the next line.
765                 */
766                 if (NULL == ws)
767                     ws = c;
768                 /*
769                 * Strip trailing tabs in literal context only;
770                 * outside, they affect the next line.
771                 */
772                 if (MDOC_LITERAL & m->flags)
773                     continue;
774                 break;
775             case '\\.':
776                 /* Skip the escaped character, too, if any. */
777                 if (c[1])
778                     c++;
779                 /* FALLTHROUGH */
780             default:
781                 ws = NULL;
782                 break;

```

```

783     }
784     end = c + 1;
785 }
786 *end = '\0';

788 if (ws)
789     mdoc_pmsg(m, line, (int)(ws-buf), MANDOCERR_EOLNSPACE);

791 if ('\0' == buf[offs] && ! (MDOC_LITERAL & m->flags)) {
792     mdoc_pmsg(m, line, (int)(c-buf), MANDOCERR_NOBLANKLN);

794     /*
795      * Insert a 'sp' in the case of a blank line. Technically,
796      * blank lines aren't allowed, but enough manuals assume this
797      * behaviour that we want to work around it.
798      */
799     if (! mdoc_elem_alloc(m, line, offs, MDOC_sp, NULL))
800         return(0);

802     m->next = MDOC_NEXT_SIBLING;
803     return(1);
804 }

806 if (! mdoc_word_alloc(m, line, offs, buf+offs))
807     return(0);

809 if (MDOC_LITERAL & m->flags)
810     return(1);

812 /*
813  * End-of-sentence check. If the last character is an unescaped
814  * EOS character, then flag the node as being the end of a
815  * sentence. The front-end will know how to interpret this.
816  */

818 assert(buf < end);

820 if (mandoc_eos(buf+offs, (size_t)(end-buf-offs), 0))
821     m->last->flags |= MDOC_EOS;

823 return(1);
824 }

827 /*
828  * Parse a macro line, that is, a line beginning with the control
829  * character.
830  */
831 static int
832 mdoc_pmacro(struct mdoc *m, int ln, char *buf, int offs)
833 {
834     enum mdoct      tok;
835     int             i, sv;
836     char            mac[5];
837     struct mdoc_node *n;

839     /* Empty post-control lines are ignored. */

841     if ('"' == buf[offs]) {
842         mdoc_pmsg(m, ln, offs, MANDOCERR_BADCOMMENT);
843         return(1);
844     } else if ('\0' == buf[offs])
845         return(1);

847     sv = offs;

```

```

849     /*
850      * Copy the first word into a nil-terminated buffer.
851      * Stop copying when a tab, space, or eoln is encountered.
852      */

854     i = 0;
855     while (i < 4 && '\0' != buf[offs] &&
856           ' ' != buf[offs] && '\t' != buf[offs])
857         mac[i++] = buf[offs++];

859     mac[i] = '\0';

861     tok = (i > 1 || i < 4) ? mdoc_hash_find(mac) : MDOC_MAX;

863     if (MDOC_MAX == tok) {
864         mdoc_vmsg(MANDOCERR_MACRO, m->parse,
865                 ln, sv, "%s", buf + sv - 1);
866         return(1);
867     }

869     /* Disregard the first trailing tab, if applicable. */

871     if ('\t' == buf[offs])
872         offs++;

874     /* Jump to the next non-whitespace word. */

876     while (buf[offs] && ' ' == buf[offs])
877         offs++;

879     /*
880      * Trailing whitespace. Note that tabs are allowed to be passed
881      * into the parser as "text", so we only warn about spaces here.
882      */

884     if ('\0' == buf[offs] && ' ' == buf[offs - 1])
885         mdoc_pmsg(m, ln, offs - 1, MANDOCERR_EOLNSPACE);

887     /*
888      * If an initial macro or a list invocation, divert directly
889      * into macro processing.
890      */

892     if (NULL == m->last || MDOC_It == tok || MDOC_El == tok) {
893         if (! mdoc_macro(m, tok, ln, sv, &offs, buf))
894             goto err;
895         return(1);
896     }

898     n = m->last;
899     assert(m->last);

901     /*
902      * If the first macro of a 'Bl -column', open an 'It' block
903      * context around the parsed macro.
904      */

906     if (MDOC_Bl == n->tok && MDOC_BODY == n->type &&
907         LIST_column == n->norm->Bl.type) {
908         m->flags |= MDOC_FREECOL;
909         if (! mdoc_macro(m, MDOC_It, ln, sv, &sv, buf))
910             goto err;
911         return(1);
912     }

914     /*

```

```

915  * If we're following a block-level 'It' within a 'Bl -column'
916  * context (perhaps opened in the above block or in ptext()),
917  * then open an 'It' block context around the parsed macro.
918  */
919
920  if (MDOC_It == n->tok && MDOC_BLOCK == n->type &&
921      NULL != n->parent &&
922      MDOC_Bl == n->parent->tok &&
923      LIST_column == n->parent->norm->Bl.type) {
924      m->flags |= MDOC_FREECOL;
925      if (!mdoc_macro(m, MDOC_It, ln, sv, &sv, buf))
926          goto err;
927      return(1);
928  }
929
930  /* Normal processing of a macro. */
931
932  if (!mdoc_macro(m, tok, ln, sv, &offs, buf))
933      goto err;
934
935  return(1);
936
937 err:  /* Error out. */
938
939  m->flags |= MDOC_HALT;
940  return(0);
941 }
942
943 enum mdelim
944 mdoc_isdelim(const char *p)
945 {
946
947  if ('\0' == p[0])
948      return(DELM_NONE);
949
950  if ('\0' == p[1])
951      switch (p[0]) {
952      case(''):
953          /* FALLTHROUGH */
954      case('['):
955          return(DELM_OPEN);
956      case('|'):
957          return(DELM_MIDDLE);
958      case('.'):
959          /* FALLTHROUGH */
960      case(','):
961          /* FALLTHROUGH */
962      case(';'):
963          /* FALLTHROUGH */
964      case(':'):
965          /* FALLTHROUGH */
966      case('?'):
967          /* FALLTHROUGH */
968      case('!'):
969          /* FALLTHROUGH */
970      case(')'):
971          /* FALLTHROUGH */
972      case(']'):
973          return(DELM_CLOSE);
974      default:
975          return(DELM_NONE);
976  }
977
978  if ('\\' != p[0])
979      return(DELM_NONE);

```

```

981  if (0 == strcmp(p + 1, "."))
982      return(DELM_CLOSE);
983  if (0 == strcmp(p + 1, "(Ba"))
984      return(DELM_MIDDLE);
985
986  return(DELM_NONE);
987 }

```

8425 Sat Jul 19 19:24:30 2014
new/usr/src/cmd/mandoc/mdoc.h
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD

```
1 /* $Id: mdoc.h,v 1.122 2011/03/22 14:05:45 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef MDOC_H
18 #define MDOC_H
20 enum mdoct {
21     MDOC_Ap = 0,
22     MDOC_Dd,
23     MDOC_Dt,
24     MDOC_Os,
25     MDOC_Sh,
26     MDOC_Ss,
27     MDOC_Pp,
28     MDOC_Dl,
29     MDOC_Dl,
30     MDOC_Bd,
31     MDOC_Ed,
32     MDOC_Bl,
33     MDOC_El,
34     MDOC_It,
35     MDOC_Ad,
36     MDOC_An,
37     MDOC_Ar,
38     MDOC_Cd,
39     MDOC_Cm,
40     MDOC_Dv,
41     MDOC_Er,
42     MDOC_Ev,
43     MDOC_Ex,
44     MDOC_Fa,
45     MDOC_Fd,
46     MDOC_Fl,
47     MDOC_Fn,
48     MDOC_Ft,
49     MDOC_Ic,
50     MDOC_In,
51     MDOC_Li,
52     MDOC_Nd,
53     MDOC_Nm,
54     MDOC_Op,
55     MDOC_Ot,
56     MDOC_Pa,
```

```
57     MDOC_Rv,
58     MDOC_St,
59     MDOC_Va,
60     MDOC_Vt,
61     MDOC_Xr,
62     MDOC_A,
63     MDOC_B,
64     MDOC_D,
65     MDOC_I,
66     MDOC_J,
67     MDOC_N,
68     MDOC_O,
69     MDOC_P,
70     MDOC_R,
71     MDOC_T,
72     MDOC_V,
73     MDOC_Ac,
74     MDOC_Ao,
75     MDOC_Aq,
76     MDOC_At,
77     MDOC_Bc,
78     MDOC_Bf,
79     MDOC_Bo,
80     MDOC_Bq,
81     MDOC_Bsx,
82     MDOC_Bx,
83     MDOC_Db,
84     MDOC_Dc,
85     MDOC_Do,
86     MDOC_Dq,
87     MDOC_Ec,
88     MDOC_Ef,
89     MDOC_Em,
90     MDOC_Eo,
91     MDOC_Fx,
92     MDOC_Ms,
93     MDOC_No,
94     MDOC_Ns,
95     MDOC_Nx,
96     MDOC_Ox,
97     MDOC_Pc,
98     MDOC_Pf,
99     MDOC_Po,
100    MDOC_Pq,
101    MDOC_Qc,
102    MDOC_Ql,
103    MDOC_Qo,
104    MDOC_Qq,
105    MDOC_Re,
106    MDOC_Rs,
107    MDOC_Sc,
108    MDOC_So,
109    MDOC_Sq,
110    MDOC_Sm,
111    MDOC_Sx,
112    MDOC_Sy,
113    MDOC_Tn,
114    MDOC_Ux,
115    MDOC_Xc,
116    MDOC_Xo,
117    MDOC_Fo,
118    MDOC_Fc,
119    MDOC_Oo,
120    MDOC_Oc,
121    MDOC_Bk,
122    MDOC_Ek,
```

```

123 MDOC_Bt,
124 MDOC_Hf,
125 MDOC_Fr,
126 MDOC_Ud,
127 MDOC_Lb,
128 MDOC_Lp,
129 MDOC_Lk,
130 MDOC_Mt,
131 MDOC_Brq,
132 MDOC_Bro,
133 MDOC_Brc,
134 MDOC_C,
135 MDOC_Es,
136 MDOC_En,
137 MDOC_Dx,
138 MDOC_Q,
139 MDOC_br,
140 MDOC_sp,
141 MDOC_U,
142 MDOC_Ta,
143 MDOC_MAX
144 };

146 enum mdocargt {
147 MDOC_Split, /* -split */
148 MDOC_Nosplit, /* -nospli */
149 MDOC_Ragged, /* -ragged */
150 MDOC_Unfilled, /* -unfilled */
151 MDOC_Literal, /* -literal */
152 MDOC_File, /* -file */
153 MDOC_Offset, /* -offset */
154 MDOC_Bullet, /* -bullet */
155 MDOC_Dash, /* -dash */
156 MDOC_Hyphen, /* -hyphen */
157 MDOC_Item, /* -item */
158 MDOC_Enum, /* -enum */
159 MDOC_Tag, /* -tag */
160 MDOC_Diag, /* -diag */
161 MDOC_Hang, /* -hang */
162 MDOC_Ohang, /* -chang */
163 MDOC_Inset, /* -inset */
164 MDOC_Column, /* -column */
165 MDOC_Width, /* -width */
166 MDOC_Compact, /* -compact */
167 MDOC_Std, /* -std */
168 MDOC_Filled, /* -filled */
169 MDOC_Words, /* -words */
170 MDOC_Emphasis, /* -emphasis */
171 MDOC_Symbolic, /* -symbolic */
172 MDOC_Nested, /* -nested */
173 MDOC_Centred, /* -centered */
174 MDOC_ARG_MAX
175 };

177 enum mdoc_type {
178 MDOC_TEXT,
179 MDOC_ELEM,
180 MDOC_HEAD,
181 MDOC_TAIL,
182 MDOC_BODY,
183 MDOC_BLOCK,
184 MDOC_TBL,
185 MDOC_EQN,
186 MDOC_ROOT
187 };

```

```

189 /*
190 * Section (named/unnamed) of 'Sh'. Note that these appear in the
191 * conventional order imposed by mdoc.7. In the case of SEC_NONE, no
192 * section has been invoked (this shouldn't happen). SEC_CUSTOM refers
193 * to other sections.
194 */
195 enum mdoc_sec {
196 SEC_NONE = 0,
197 SEC_NAME, /* NAME */
198 SEC_LIBRARY, /* LIBRARY */
199 SEC_SYNOPSIS, /* SYNOPSIS */
200 SEC_DESCRIPTION, /* DESCRIPTION */
201 SEC_IMPLEMENTATION, /* IMPLEMENTATION NOTES */
202 SEC_RETURN_VALUES, /* RETURN VALUES */
203 SEC_ENVIRONMENT, /* ENVIRONMENT */
204 SEC_FILES, /* FILES */
205 SEC_EXIT_STATUS, /* EXIT STATUS */
206 SEC_EXAMPLES, /* EXAMPLES */
207 SEC_DIAGNOSTICS, /* DIAGNOSTICS */
208 SEC_COMPATIBILITY, /* COMPATIBILITY */
209 SEC_ERRORS, /* ERRORS */
210 SEC_SEE_ALSO, /* SEE ALSO */
211 SEC_STANDARDS, /* STANDARDS */
212 SEC_HISTORY, /* HISTORY */
213 SEC_AUTHORS, /* AUTHORS */
214 SEC_CAVEATS, /* CAVEATS */
215 SEC_BUGS, /* BUGS */
216 SEC_SECURITY, /* SECURITY */
217 SEC_CUSTOM,
218 SEC_MAX
219 };

221 struct mdoc_meta {
222 char *msec; /* 'Dt' section (1, 3p, etc.) */
223 char *vol; /* 'Dt' volume (implied) */
224 char *arch; /* 'Dt' arch (i386, etc.) */
225 char *date; /* 'Dd' normalised date */
226 char *title; /* 'Dt' title (FOO, etc.) */
227 char *os; /* 'Os' system (OpenBSD, etc.) */
228 char *name; /* leading 'Nm' name */
229 };

231 /*
232 * An argument to a macro (multiple values = '-column xxx yyy').
233 */
234 struct mdoc_argv {
235 enum mdocargt arg; /* type of argument */
236 int line;
237 int pos;
238 size_t sz; /* elements in "value" */
239 char **value; /* argument strings */
240 };

242 /*
243 * Reference-counted macro arguments. These are refcounted because
244 * blocks have multiple instances of the same arguments spread across
245 * the HEAD, BODY, TAIL, and BLOCK node types.
246 */
247 struct mdoc_arg {
248 size_t argc;
249 struct mdoc_argv *argv;
250 unsigned int refcnt;
251 };

253 /*
254 * Indicates that a BODY's formatting has ended, but the scope is still

```

```

255 * open. Used for syntax-broken blocks.
256 */
257 enum mdoc_endbody {
258 ENDBODY_NOT = 0,
259 ENDBODY_SPACE, /* is broken: append a space */
260 ENDBODY_NOSPACE /* is broken: don't append a space */
261 };

263 enum mdoc_list {
264 LIST_NONE = 0,
265 LIST_bullet, /* -bullet */
266 LIST_column, /* -column */
267 LIST_dash, /* -dash */
268 LIST_diag, /* -diag */
269 LIST_enum, /* -enum */
270 LIST_hang, /* -hang */
271 LIST_hyphen, /* -hyphen */
272 LIST_inset, /* -inset */
273 LIST_item, /* -item */
274 LIST_chang, /* -chang */
275 LIST_tag, /* -tag */
276 LIST_MAX
277 };

279 enum mdoc_disp {
280 DISP_NONE = 0,
281 DISP_centred, /* -centered */
282 DISP_ragged, /* -ragged */
283 DISP_unfilled, /* -unfilled */
284 DISP_filled, /* -filled */
285 DISP_literal /* -literal */
286 };

288 enum mdoc_auth {
289 AUTH_NONE = 0,
290 AUTH_split, /* -split */
291 AUTH_nosplit /* -nosplit */
292 };

294 enum mdoc_font {
295 FONT_NONE = 0,
296 FONT_Em, /* Em, -emphasis */
297 FONT_Li, /* Li, -literal */
298 FONT_Sy /* Sy, -symbolic */
299 };

301 struct mdoc_bd {
302 const char *offs; /* -offset */
303 enum mdoc_disp type; /* -ragged, etc. */
304 int comp; /* -compact */
305 };

307 struct mdoc_bl {
308 const char *width; /* -width */
309 const char *offs; /* -offset */
310 enum mdoc_list type; /* -tag, -enum, etc. */
311 int comp; /* -compact */
312 size_t ncols; /* -column arg count */
313 const char **cols; /* -column val ptr */
314 };

316 struct mdoc_bf {
317 enum mdoc_font font; /* font */
318 };

320 struct mdoc_an {

```

```

321 enum mdoc_auth auth; /* -split, etc. */
322 };

324 struct mdoc_rs {
325 int quote_T; /* whether to quote %T */
326 };

328 /*
329 * Consists of normalised node arguments. These should be used instead
330 * of iterating through the mdoc_arg pointers of a node: defaults are
331 * provided, etc.
332 */
333 union mdoc_data {
334 struct mdoc_an An;
335 struct mdoc_bd Bd;
336 struct mdoc_bf Bf;
337 struct mdoc_bl Bl;
338 struct mdoc_rs Rs;
339 };

341 /*
342 * Single node in tree-linked AST.
343 */
344 struct mdoc_node {
345 struct mdoc_node *parent; /* parent AST node */
346 struct mdoc_node *child; /* first child AST node */
347 struct mdoc_node *last; /* last child AST node */
348 struct mdoc_node *next; /* sibling AST node */
349 struct mdoc_node *prev; /* prior sibling AST node */
350 int nchild; /* number children */
351 int line; /* parse line */
352 int pos; /* parse column */
353 enum mdoct tok; /* tok or MDOC_MAX if none */
354 int flags;
355 #define MDOC_VALID (1 << 0) /* has been validated */
356 #define MDOC_EOS (1 << 2) /* at sentence boundary */
357 #define MDOC_LINE (1 << 3) /* first macro/text on line */
358 #define MDOC_SYNPRETTY (1 << 4) /* SYNOPSIS-style formatting */
359 #define MDOC_ENDED (1 << 5) /* rendering has been ended */
360 #define MDOC_DELIMO (1 << 6)
361 #define MDOC_DELMC (1 << 7)
362 enum mdoc_type type; /* AST node type */
363 enum mdoc_sec sec; /* current named section */
364 union mdoc_data *norm; /* normalised args */
365 /* FIXME: these can be union'd to shave a few bytes. */
366 struct mdoc_arg *args; /* BLOCK/ELEM */
367 struct mdoc_node *pending; /* BLOCK */
368 struct mdoc_node *head; /* BLOCK */
369 struct mdoc_node *body; /* BLOCK */
370 struct mdoc_node *tail; /* BLOCK */
371 char *string; /* TEXT */
372 const struct tbl_span *span; /* TBL */
373 const struct eqn *eqn; /* EQN */
374 enum mdoc_endbody end; /* BODY */
375 };

377 /* Names of macros. Index is enum mdoct. */
378 extern const char *const *mdoc_macronames;

380 /* Names of macro args. Index is enum mdocargt. */
381 extern const char *const *mdoc_argnames;

383 __BEGIN_DECLS

385 struct mdoc;

```

new/usr/src/cmd/mandoc/mdoc.h

7

```
387 const struct mdoc_node *mdoc_node(const struct mdoc *);  
388 const struct mdoc_meta *mdoc_meta(const struct mdoc *);  
  
390 __END_DECLS  
  
392 #endif /*!MDOC_H*/
```



```

*****
17101 Sat Jul 19 19:24:30 2014
new/usr/src/cmd/mandoc/mdoc_argv.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: mdoc_argv.c,v 1.82 2012/03/23 05:50:24 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <sys/types.h>
22
23 #include <assert.h>
24 #include <stdlib.h>
25 #include <stdio.h>
26 #include <string.h>
27
28 #include "mdoc.h"
29 #include "mandoc.h"
30 #include "libmdoc.h"
31 #include "libmandoc.h"
32
33 #define MULTI_STEP      5 /* pre-allocate argument values */
34 #define DELIMSZ        6 /* max possible size of a delimiter */
35
36 enum   argsflag {
37     ARGSFL_NONE = 0,
38     ARGSFL_DELIM, /* handle delimiters of [[:delim:][ ]]+ */
39     ARGSFL_TABSEP /* handle tab/'Ta' separated phrases */
40 };
41
42 enum   argvflag {
43     ARGV_NONE, /* no args to flag (e.g., -split) */
44     ARGV_SINGLE, /* one arg to flag (e.g., -file xxx) */
45     ARGV_MULTI, /* multiple args (e.g., -column xxx yyy) */
46     ARGV_OPT_SINGLE /* optional arg (e.g., -offset [xxx]) */
47 };
48
49 struct mdocarg {
50     enum argsflag   flags;
51     const enum mdocargt *argvs;
52 };
53
54 static void      argn_free(struct mdoc_arg *, int);
55 static enum margserr  args(struct mdoc *, int, int *,
56                             char *, enum argsflag, char **);

```

```

57 static int      args_checkpunct(const char *, int);
58 static int      argv_multi(struct mdoc *, int,
59                             struct mdoc_argv *, int *, char *);
60 static int      argv_opt_single(struct mdoc *, int,
61                                 struct mdoc_argv *, int *, char *);
62 static int      argv_single(struct mdoc *, int,
63                              struct mdoc_argv *, int *, char *);
64
65 static const enum argvflag argvflags[MDOC_ARG_MAX] = {
66     ARGV_NONE, /* MDOC_Split */
67     ARGV_NONE, /* MDOC_Nosplit */
68     ARGV_NONE, /* MDOC_Ragged */
69     ARGV_NONE, /* MDOC_Unfilled */
70     ARGV_NONE, /* MDOC_Literal */
71     ARGV_SINGLE, /* MDOC_File */
72     ARGV_OPT_SINGLE, /* MDOC_Offset */
73     ARGV_NONE, /* MDOC_Bullet */
74     ARGV_NONE, /* MDOC_Dash */
75     ARGV_NONE, /* MDOC_Hyphen */
76     ARGV_NONE, /* MDOC_Item */
77     ARGV_NONE, /* MDOC_Enum */
78     ARGV_NONE, /* MDOC_Tag */
79     ARGV_NONE, /* MDOC_Diag */
80     ARGV_NONE, /* MDOC_Hang */
81     ARGV_NONE, /* MDOC_Ohang */
82     ARGV_NONE, /* MDOC_Inset */
83     ARGV_MULTI, /* MDOC_Column */
84     ARGV_OPT_SINGLE, /* MDOC_Width */
85     ARGV_NONE, /* MDOC_Compact */
86     ARGV_NONE, /* MDOC_Std */
87     ARGV_NONE, /* MDOC_Filled */
88     ARGV_NONE, /* MDOC_Words */
89     ARGV_NONE, /* MDOC_Emphasis */
90     ARGV_NONE, /* MDOC_Symbolic */
91     ARGV_NONE /* MDOC_Symbolic */
92 };
93
94 static const enum mdocargt args_Ex[] = {
95     MDOC_Std,
96     MDOC_ARG_MAX
97 };
98
99 static const enum mdocargt args_An[] = {
100     MDOC_Split,
101     MDOC_Nosplit,
102     MDOC_ARG_MAX
103 };
104
105 static const enum mdocargt args_Bd[] = {
106     MDOC_Ragged,
107     MDOC_Unfilled,
108     MDOC_Filled,
109     MDOC_Literal,
110     MDOC_File,
111     MDOC_Offset,
112     MDOC_Compact,
113     MDOC_Centred,
114     MDOC_ARG_MAX
115 };
116
117 static const enum mdocargt args_Bf[] = {
118     MDOC_Emphasis,
119     MDOC_Literal,
120     MDOC_Symbolic,
121     MDOC_ARG_MAX
122 };

```

```

124 static const enum mdocargt args_Bk[] = {
125     MDOC_Words,
126     MDOC_ARG_MAX
127 };

129 static const enum mdocargt args_Bl[] = {
130     MDOC_Bullet,
131     MDOC_Dash,
132     MDOC_Hyphen,
133     MDOC_Item,
134     MDOC_Enum,
135     MDOC_Tag,
136     MDOC_Diag,
137     MDOC_Hang,
138     MDOC_Ohang,
139     MDOC_Inset,
140     MDOC_Column,
141     MDOC_Width,
142     MDOC_Offset,
143     MDOC_Compact,
144     MDOC_Nested,
145     MDOC_ARG_MAX
146 };

148 static const struct mdocarg mdocargs[MDOC_MAX] = {
149     { ARGSFL_NONE, NULL }, /* Ap */
150     { ARGSFL_NONE, NULL }, /* Dd */
151     { ARGSFL_NONE, NULL }, /* Dt */
152     { ARGSFL_NONE, NULL }, /* Os */
153     { ARGSFL_NONE, NULL }, /* Sh */
154     { ARGSFL_NONE, NULL }, /* Ss */
155     { ARGSFL_NONE, NULL }, /* Pp */
156     { ARGSFL_DELIM, NULL }, /* D1 */
157     { ARGSFL_DELIM, NULL }, /* D1 */
158     { ARGSFL_NONE, args_Bd }, /* Bd */
159     { ARGSFL_NONE, NULL }, /* Ed */
160     { ARGSFL_NONE, args_Bl }, /* Bl */
161     { ARGSFL_NONE, NULL }, /* El */
162     { ARGSFL_NONE, NULL }, /* It */
163     { ARGSFL_DELIM, NULL }, /* Ad */
164     { ARGSFL_DELIM, args_An }, /* An */
165     { ARGSFL_DELIM, NULL }, /* Ar */
166     { ARGSFL_NONE, NULL }, /* Cd */
167     { ARGSFL_DELIM, NULL }, /* Cm */
168     { ARGSFL_DELIM, NULL }, /* Dv */
169     { ARGSFL_DELIM, NULL }, /* Er */
170     { ARGSFL_DELIM, NULL }, /* Ev */
171     { ARGSFL_NONE, args_Ex }, /* Ex */
172     { ARGSFL_DELIM, NULL }, /* Fa */
173     { ARGSFL_NONE, NULL }, /* Fd */
174     { ARGSFL_DELIM, NULL }, /* Fl */
175     { ARGSFL_DELIM, NULL }, /* Fn */
176     { ARGSFL_DELIM, NULL }, /* Ft */
177     { ARGSFL_DELIM, NULL }, /* Ic */
178     { ARGSFL_NONE, NULL }, /* In */
179     { ARGSFL_DELIM, NULL }, /* Li */
180     { ARGSFL_NONE, NULL }, /* Nd */
181     { ARGSFL_DELIM, NULL }, /* Nm */
182     { ARGSFL_DELIM, NULL }, /* Op */
183     { ARGSFL_NONE, NULL }, /* Ot */
184     { ARGSFL_DELIM, NULL }, /* Pa */
185     { ARGSFL_NONE, args_Ex }, /* Rv */
186     { ARGSFL_DELIM, NULL }, /* St */
187     { ARGSFL_DELIM, NULL }, /* Va */
188     { ARGSFL_DELIM, NULL }, /* Vt */

```

```

189     { ARGSFL_DELIM, NULL }, /* Xr */
190     { ARGSFL_NONE, NULL }, /* %A */
191     { ARGSFL_NONE, NULL }, /* %B */
192     { ARGSFL_NONE, NULL }, /* %D */
193     { ARGSFL_NONE, NULL }, /* %I */
194     { ARGSFL_NONE, NULL }, /* %J */
195     { ARGSFL_NONE, NULL }, /* %N */
196     { ARGSFL_NONE, NULL }, /* %O */
197     { ARGSFL_NONE, NULL }, /* %P */
198     { ARGSFL_NONE, NULL }, /* %R */
199     { ARGSFL_NONE, NULL }, /* %T */
200     { ARGSFL_NONE, NULL }, /* %V */
201     { ARGSFL_DELIM, NULL }, /* Ac */
202     { ARGSFL_NONE, NULL }, /* Ao */
203     { ARGSFL_DELIM, NULL }, /* Aq */
204     { ARGSFL_DELIM, NULL }, /* At */
205     { ARGSFL_DELIM, NULL }, /* Bc */
206     { ARGSFL_NONE, args_Bf }, /* Bf */
207     { ARGSFL_NONE, NULL }, /* Bo */
208     { ARGSFL_DELIM, NULL }, /* Bq */
209     { ARGSFL_DELIM, NULL }, /* Bsx */
210     { ARGSFL_DELIM, NULL }, /* Bx */
211     { ARGSFL_NONE, NULL }, /* Db */
212     { ARGSFL_DELIM, NULL }, /* Dc */
213     { ARGSFL_NONE, NULL }, /* Do */
214     { ARGSFL_DELIM, NULL }, /* Dq */
215     { ARGSFL_DELIM, NULL }, /* Ec */
216     { ARGSFL_NONE, NULL }, /* Ef */
217     { ARGSFL_DELIM, NULL }, /* Em */
218     { ARGSFL_NONE, NULL }, /* Eo */
219     { ARGSFL_DELIM, NULL }, /* Fx */
220     { ARGSFL_DELIM, NULL }, /* Ms */
221     { ARGSFL_DELIM, NULL }, /* No */
222     { ARGSFL_DELIM, NULL }, /* Ns */
223     { ARGSFL_DELIM, NULL }, /* Nx */
224     { ARGSFL_DELIM, NULL }, /* Ox */
225     { ARGSFL_DELIM, NULL }, /* Pc */
226     { ARGSFL_DELIM, NULL }, /* Pf */
227     { ARGSFL_NONE, NULL }, /* Po */
228     { ARGSFL_DELIM, NULL }, /* Pq */
229     { ARGSFL_DELIM, NULL }, /* Qc */
230     { ARGSFL_DELIM, NULL }, /* Ql */
231     { ARGSFL_NONE, NULL }, /* Qo */
232     { ARGSFL_DELIM, NULL }, /* Qq */
233     { ARGSFL_NONE, NULL }, /* Re */
234     { ARGSFL_NONE, NULL }, /* Rs */
235     { ARGSFL_DELIM, NULL }, /* Sc */
236     { ARGSFL_NONE, NULL }, /* So */
237     { ARGSFL_DELIM, NULL }, /* Sq */
238     { ARGSFL_NONE, NULL }, /* Sm */
239     { ARGSFL_DELIM, NULL }, /* Sx */
240     { ARGSFL_DELIM, NULL }, /* Sy */
241     { ARGSFL_DELIM, NULL }, /* Tn */
242     { ARGSFL_DELIM, NULL }, /* Ux */
243     { ARGSFL_DELIM, NULL }, /* Xc */
244     { ARGSFL_NONE, NULL }, /* Xo */
245     { ARGSFL_NONE, NULL }, /* Fo */
246     { ARGSFL_NONE, NULL }, /* Fc */
247     { ARGSFL_NONE, NULL }, /* Oo */
248     { ARGSFL_DELIM, NULL }, /* Oc */
249     { ARGSFL_NONE, args_Bk }, /* Bk */
250     { ARGSFL_NONE, NULL }, /* Ek */
251     { ARGSFL_NONE, NULL }, /* Bt */
252     { ARGSFL_NONE, NULL }, /* Hf */
253     { ARGSFL_NONE, NULL }, /* Fr */
254     { ARGSFL_NONE, NULL }, /* Ud */

```

```

255 { ARGSFL_NONE, NULL }, /* Lb */
256 { ARGSFL_NONE, NULL }, /* Lp */
257 { ARGSFL_DELM, NULL }, /* Lk */
258 { ARGSFL_DELM, NULL }, /* Mt */
259 { ARGSFL_DELM, NULL }, /* Brq */
260 { ARGSFL_NONE, NULL }, /* Bro */
261 { ARGSFL_DELM, NULL }, /* Brc */
262 { ARGSFL_NONE, NULL }, /* %C */
263 { ARGSFL_NONE, NULL }, /* Es */
264 { ARGSFL_NONE, NULL }, /* En */
265 { ARGSFL_NONE, NULL }, /* Dx */
266 { ARGSFL_NONE, NULL }, /* %Q */
267 { ARGSFL_NONE, NULL }, /* br */
268 { ARGSFL_NONE, NULL }, /* sp */
269 { ARGSFL_NONE, NULL }, /* %U */
270 { ARGSFL_NONE, NULL }, /* Ta */
271 };

274 /*
275  * Parse an argument from line text. This comes in the form of -key
276  * [value0...], which may either have a single mandatory value, at least
277  * one mandatory value, an optional single value, or no value.
278  */
279 enum margverr
280 mdoc_argv(struct mdoc *m, int line, enum mdoct tok,
281          struct mdoc_arg **v, int *pos, char *buf)
282 {
283     char          *p, sv;
284     struct mdoc_arg tmp;
285     struct mdoc_arg *arg;
286     const enum mdocargt *ap;

288     if ('\0' == buf[*pos])
289         return(ARGV_EOLN);
290     else if (NULL == (ap = mdocargs[tok].argvs))
291         return(ARGV_WORD);
292     else if ('-' != buf[*pos])
293         return(ARGV_WORD);

295     /* Seek to the first unescaped space. */

297     p = &buf[++(*pos)];

299     assert(*pos > 0);

301     for ( ; buf[*pos] ; (*pos)++)
302         if (' ' == buf[*pos] && '\\\ ' != buf[*pos - 1])
303             break;

305     /*
306     * We want to nil-terminate the word to look it up (it's easier
307     * that way). But we may not have a flag, in which case we need
308     * to restore the line as-is. So keep around the stray byte,
309     * which we'll reset upon exiting (if necessary).
310     */

312     if ('\0' != (sv = buf[*pos]))
313         buf[(++*pos)] = '\0';

315     /*
316     * Now look up the word as a flag. Use temporary storage that
317     * we'll copy into the node's flags, if necessary.
318     */

320     memset(&tmp, 0, sizeof(struct mdoc_arg));

```

```

322     tmp.line = line;
323     tmp.pos = *pos;
324     tmp.arg = MDOC_ARG_MAX;

326     while (MDOC_ARG_MAX != (tmp.arg = *ap++))
327         if (0 == strcmp(p, mdoc_argnames[tmp.arg]))
328             break;

330     if (MDOC_ARG_MAX == tmp.arg) {
331         /*
332          * The flag was not found.
333          * Restore saved zeroed byte and return as a word.
334          */
335         if (sv)
336             buf[*pos - 1] = sv;
337         return(ARGV_WORD);
338     }

340     /* Read to the next word (the argument). */

342     while (buf[*pos] && ' ' == buf[*pos])
343         (*pos)++;

345     switch (argvflags[tmp.arg]) {
346     case (ARGV_SINGLE):
347         if ( ! argv_single(m, line, &tmp, pos, buf))
348             return(ARGV_ERROR);
349         break;
350     case (ARGV_MULT):
351         if ( ! argv_multi(m, line, &tmp, pos, buf))
352             return(ARGV_ERROR);
353         break;
354     case (ARGV_OPT_SINGLE):
355         if ( ! argv_opt_single(m, line, &tmp, pos, buf))
356             return(ARGV_ERROR);
357         break;
358     case (ARGV_NONE):
359         break;
360     }

362     if (NULL == (arg = *v))
363         arg = *v = mandoc_calloc(1, sizeof(struct mdoc_arg));

365     arg->argc++;
366     arg->argv = mandoc_realloc
367         (arg->argv, arg->argc * sizeof(struct mdoc_arg));

369     memcpy(&arg->argv[(int)arg->argc - 1],
370          &tmp, sizeof(struct mdoc_arg));

372     return(ARGV_ARG);
373 }

375 void
376 mdoc_argv_free(struct mdoc_arg *p)
377 {
378     int          i;

380     if (NULL == p)
381         return;

383     if (p->refcnt) {
384         --(p->refcnt);
385         if (p->refcnt)
386             return;

```

```

387     }
388     assert(p->argc);

390     for (i = (int)p->argc - 1; i >= 0; i--)
391         argn_free(p, i);

393     free(p->argv);
394     free(p);
395 }

397 static void
398 argn_free(struct mdoc_arg *p, int iarg)
399 {
400     struct mdoc_argv *arg;
401     int j;

403     arg = &p->argv[iarg];

405     if (arg->sz && arg->value) {
406         for (j = (int)arg->sz - 1; j >= 0; j--)
407             free(arg->value[j]);
408         free(arg->value);
409     }

411     for (--p->argc; iarg < (int)p->argc; iarg++)
412         p->argv[iarg] = p->argv[iarg+1];
413 }

415 enum margserr
416 mdoc_zargs(struct mdoc *m, int line, int *pos, char *buf, char **v)
417 {
419     return(args(m, line, pos, buf, ARGSFL_NONE, v));
420 }

422 enum margserr
423 mdoc_args(struct mdoc *m, int line, int *pos,
424           char *buf, enum mdoct tok, char **v)
425 {
426     enum argsflag fl;
427     struct mdoc_node *n;

429     fl = mdocargs[tok].flags;

431     if (MDOC_It != tok)
432         return(args(m, line, pos, buf, fl, v));

434     /*
435      * We know that we're in an 'It', so it's reasonable to expect
436      * us to be sitting in a 'Bl'. Someday this may not be the case
437      * (if we allow random 'It's sitting out there), so provide a
438      * safe fall-back into the default behaviour.
439      */

441     for (n = m->last; n; n = n->parent)
442         if (MDOC_Bl == n->tok)
443             if (LIST_column == n->norm->Bl.type) {
444                 fl = ARGSFL_TABSEP;
445                 break;
446             }

448     return(args(m, line, pos, buf, fl, v));
449 }

451 static enum margserr
452 args(struct mdoc *m, int line, int *pos,

```

```

453     char *buf, enum argsflag fl, char **v)
454 {
455     char *p, *pp;
456     enum margserr rc;

458     if ('\0' == buf[*pos]) {
459         if (MDOC_PPHRASE & m->flags)
460             return(ARGS_EOLN);
461         /*
462          * If we're not in a partial phrase and the flag for
463          * being a phrase literal is still set, the punctuation
464          * is unterminated.
465          */
466         if (MDOC_PHRASELIT & m->flags)
467             mdoc_pmsg(m, line, *pos, MANDOCERR_BADQUOTE);

469         m->flags &= ~MDOC_PHRASELIT;
470         return(ARGS_EOLN);
471     }

473     *v = &buf[*pos];

475     if (ARGSFL_DELIM == fl)
476         if (args_checkpunct(buf, *pos))
477             return(ARGS_PUNCT);

479     /*
480      * First handle TABSEP items, restricted to 'Bl -column'. This
481      * ignores conventional token parsing and instead uses tabs or
482      * 'Ta' macros to separate phrases. Phrases are parsed again
483      * for arguments at a later phase.
484      */

486     if (ARGSFL_TABSEP == fl) {
487         /* Scan ahead to tab (can't be escaped). */
488         p = strchr(*v, '\t');
489         pp = NULL;

491         /* Scan ahead to unescaped 'Ta'. */
492         if ( ! (MDOC_PHRASELIT & m->flags) )
493             for (pp = *v; ; pp++) {
494                 if (NULL == (pp = strstr(pp, "Ta")))
495                     break;
496                 if (pp > *v && ' ' != *(pp - 1))
497                     continue;
498                 if (' ' == *(pp + 2) || '\0' == *(pp + 2))
499                     break;
500             }

502         /* By default, assume a phrase. */
503         rc = ARGS_PHRASE;

505         /*
506          * Adjust new-buffer position to be beyond delimiter
507          * mark (e.g., Ta -> end + 2).
508          */
509         if (p && pp) {
510             *pos += pp < p ? 2 : 1;
511             rc = pp < p ? ARGS_PHRASE : ARGS_PPHRASE;
512             p = pp < p ? pp : p;
513         } else if (p && ! pp) {
514             rc = ARGS_PPHRASE;
515             *pos += 1;
516         } else if (pp && ! p) {
517             p = pp;
518             *pos += 2;

```

```

519     } else {
520         rc = ARGS_PEND;
521         p = strchr(*v, 0);
522     }

524     /* Whitespace check for eoln case... */
525     if ('\0' == *p && ' ' == *(p - 1))
526         mdoc_pmsg(m, line, *pos, MANDOCERR_EOLNSPACE);

528     *pos += (int)(p - *v);

530     /* Strip delimiter's preceding whitespace. */
531     pp = p - 1;
532     while (pp > *v && ' ' == *pp) {
533         if (pp > *v && '\\\'' == *(pp - 1))
534             break;
535         pp--;
536     }
537     *(pp + 1) = 0;

539     /* Strip delimiter's proceeding whitespace. */
540     for (pp = &buf[*pos]; ' ' == *pp; pp++, (*pos)++)
541         /* Skip ahead. */ ;

543     return(rc);
544 }

546 /*
547  * Process a quoted literal. A quote begins with a double-quote
548  * and ends with a double-quote NOT preceded by a double-quote.
549  * Whitespace is NOT involved in literal termination.
550  */

552 if (MDOC_PHRASELIT & m->flags || '\\\'' == buf[*pos]) {
553     if ( ! (MDOC_PHRASELIT & m->flags))
554         *v = &buf[++(*pos)];

556     if (MDOC_PPHRASE & m->flags)
557         m->flags |= MDOC_PHRASELIT;

559     for ( ; buf[*pos]; (*pos)++) {
560         if ('\\\'' != buf[*pos])
561             continue;
562         if ('\\\'' != buf[*pos + 1])
563             break;
564         (*pos)++;
565     }

567     if ('\0' == buf[*pos]) {
568         if (MDOC_PPHRASE & m->flags)
569             return(ARGS_QWORD);
570         mdoc_pmsg(m, line, *pos, MANDOCERR_BADQUOTE);
571         return(ARGS_QWORD);
572     }

574     m->flags &= ~MDOC_PHRASELIT;
575     buf[( *pos )++] = '\0';

577     if ('\0' == buf[*pos])
578         return(ARGS_QWORD);

580     while ( ' ' == buf[*pos] )
581         (*pos)++;

583     if ('\0' == buf[*pos])
584         mdoc_pmsg(m, line, *pos, MANDOCERR_EOLNSPACE);

```

```

586         return(ARGS_QWORD);
587     }

589     p = &buf[*pos];
590     *v = mandoc_getarg(m->parse, &p, line, pos);

592     return(ARGS_WORD);
593 }

595 /*
596  * Check if the string consists only of space-separated closing
597  * delimiters. This is a bit of a dance: the first must be a close
598  * delimiter, but it may be followed by middle delimiters. Arbitrary
599  * whitespace may separate these tokens.
600  */
601 static int
602 args_checkpunct(const char *buf, int i)
603 {
604     int j;
605     char dbuf[DELIMSZ];
606     enum mdelim d;

608     /* First token must be a close-delimiter. */

610     for (j = 0; buf[i] && ' ' != buf[i] && j < DELIMSZ; j++, i++)
611         dbuf[j] = buf[i];

613     if (DELIMSZ == j)
614         return(0);

616     dbuf[j] = '\0';
617     if (DELIM_CLOSE != mdoc_isdelim(dbuf))
618         return(0);

620     while ( ' ' == buf[i] )
621         i++;

623     /* Remaining must NOT be open/none. */
624
625     while (buf[i]) {
626         j = 0;
627         while (buf[i] && ' ' != buf[i] && j < DELIMSZ)
628             dbuf[j++] = buf[i++];

630         if (DELIMSZ == j)
631             return(0);

633         dbuf[j] = '\0';
634         d = mdoc_isdelim(dbuf);
635         if (DELIM_NONE == d || DELIM_OPEN == d)
636             return(0);

638         while ( ' ' == buf[i] )
639             i++;
640     }

642     return('\0' == buf[i]);
643 }

645 static int
646 argv_multi(struct mdoc *m, int line,
647            struct mdoc_argv *v, int *pos, char *buf)
648 {
649     enum margserr ac;
650     char *p;

```

```

652     for (v->sz = 0; ; v->sz++) {
653         if ('-' == buf[*pos])
654             break;
655         ac = args(m, line, pos, buf, ARGSFL_NONE, &p);
656         if (ARGS_ERROR == ac)
657             return(0);
658         else if (ARGS_EOLN == ac)
659             break;
661
662         if (0 == v->sz % MULTI_STEP)
663             v->value = mandoc_realloc(v->value,
664                                     (v->sz + MULTI_STEP) * sizeof(char *));
665         v->value[(int)v->sz] = mandoc_strdup(p);
666     }
668     return(1);
669 }
671 static int
672 argv_opt_single(struct mdoc *m, int line,
673                struct mdoc_argv *v, int *pos, char *buf)
674 {
675     enum margserr  ac;
676     char          *p;
678     if ('-' == buf[*pos])
679         return(1);
681     ac = args(m, line, pos, buf, ARGSFL_NONE, &p);
682     if (ARGS_ERROR == ac)
683         return(0);
684     if (ARGS_EOLN == ac)
685         return(1);
687     v->sz = 1;
688     v->value = mandoc_malloc(sizeof(char *));
689     v->value[0] = mandoc_strdup(p);
691     return(1);
692 }
694 static int
695 argv_single(struct mdoc *m, int line,
696             struct mdoc_argv *v, int *pos, char *buf)
697 {
698     int          ppos;
699     enum margserr  ac;
700     char          *p;
702     ppos = *pos;
704     ac = args(m, line, pos, buf, ARGSFL_NONE, &p);
705     if (ARGS_EOLN == ac) {
706         mdoc_pmsg(m, line, ppos, MANDOCERR_SYNTARGVCOUNT);
707         return(0);
708     } else if (ARGS_ERROR == ac)
709         return(0);
711     v->sz = 1;
712     v->value = mandoc_malloc(sizeof(char *));
713     v->value[0] = mandoc_strdup(p);
715     return(1);
716 }

```

new/usr/src/cmd/mandoc/mdoc_hash.c

1

```
*****
2287 Sat Jul 19 19:24:31 2014
new/usr/src/cmd/mandoc/mdoc_hash.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: mdoc_hash.c,v 1.18 2011/07/24 18:15:14 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <sys/types.h>
22
23 #include <assert.h>
24 #include <ctype.h>
25 #include <limits.h>
26 #include <stdlib.h>
27 #include <stdio.h>
28 #include <string.h>
29
30 #include "mdoc.h"
31 #include "mandoc.h"
32 #include "libmdoc.h"
33
34 static unsigned char  table[27 * 12];
35
36 /*
37  * XXX - this hash has global scope, so if intended for use as a library
38  * with multiple callers, it will need re-invocation protection.
39  */
40 void
41 mdoc_hash_init(void)
42 {
43     int            i, j, major;
44     const char     *p;
45
46     memset(table, UCHAR_MAX, sizeof(table));
47
48     for (i = 0; i < (int)MDOC_MAX; i++) {
49         p = mdoc_macronames[i];
50
51         if (isalpha((unsigned char)p[1]))
52             major = 12 * (tolower((unsigned char)p[1]) - 97);
53         else
54             major = 12 * 26;
55
56         for (j = 0; j < 12; j++)
```

new/usr/src/cmd/mandoc/mdoc_hash.c

2

```
57             if (UCHAR_MAX == table[major + j]) {
58                 table[major + j] = (unsigned char)i;
59                 break;
60             }
61
62         assert(j < 12);
63     }
64 }
65
66 enum mdoct
67 mdoc_hash_find(const char *p)
68 {
69     int            major, i, j;
70
71     if (0 == p[0])
72         return(MDOC_MAX);
73     if (! isalpha((unsigned char)p[0]) && '%' != p[0])
74         return(MDOC_MAX);
75
76     if (isalpha((unsigned char)p[1]))
77         major = 12 * (tolower((unsigned char)p[1]) - 97);
78     else if ('1' == p[1])
79         major = 12 * 26;
80     else
81         return(MDOC_MAX);
82
83     if (p[2] && p[3])
84         return(MDOC_MAX);
85
86     for (j = 0; j < 12; j++) {
87         if (UCHAR_MAX == (i = table[major + j]))
88             break;
89         if (0 == strcmp(p, mdoc_macronames[i]))
90             return((enum mdoct)i);
91     }
92
93     return(MDOC_MAX);
94 }
```

```
*****
```

```
43911 Sat Jul 19 19:24:31 2014
```

```
new/usr/src/cmd/mandoc/mdoc_html.c
```

```
5025 import and use mandoc
```

```
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
```

```
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
```

```
Reviewed by: Robert Mustacchi <rm@joyent.com>
```

```
Reviewed by: Albert Lee <trisk@nexenta.com>
```

```
Approved by: TBD
```

```
*****
```

```
1 /* $Id: mdoc_html.c,v 1.182 2011/11/03 20:37:00 schwarze Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <sys/types.h>
22
23 #include <assert.h>
24 #include <ctype.h>
25 #include <stdio.h>
26 #include <stdlib.h>
27 #include <string.h>
28 #include <unistd.h>
29
30 #include "mandoc.h"
31 #include "out.h"
32 #include "html.h"
33 #include "mdoc.h"
34 #include "main.h"
35
36 #define INDENT          5
37
38 #define MDOC_ARGS      const struct mdoc_meta *m, \
39                       const struct mdoc_node *n, \
40                       struct html *h
41
42 #ifndef MIN
43 #define MIN(a,b)      ((*CONSTCOND*/(a)<(b))?(a):(b))
44 #endif
45
46 struct htmlmdoc {
47     int      (*pre)(MDOC_ARGS);
48     void     (*post)(MDOC_ARGS);
49 };
50
51 static void      print_mdoc(MDOC_ARGS);
52 static void      print_mdoc_head(MDOC_ARGS);
53 static void      print_mdoc_node(MDOC_ARGS);
54 static void      print_mdoc_nodelist(MDOC_ARGS);
55 static void      synopsis_pre(struct html *,
56                               const struct mdoc_node *);
```

```
58 static void      a2width(const char *, struct roffsu *);
59 static void      a2offs(const char *, struct roffsu *);
60
61 static void      mdoc_root_post(MDOC_ARGS);
62 static int       mdoc_root_pre(MDOC_ARGS);
63
64 static void      mdoc__x_post(MDOC_ARGS);
65 static int       mdoc__x_pre(MDOC_ARGS);
66 static int       mdoc_ad_pre(MDOC_ARGS);
67 static int       mdoc_an_pre(MDOC_ARGS);
68 static int       mdoc_ap_pre(MDOC_ARGS);
69 static int       mdoc_ar_pre(MDOC_ARGS);
70 static int       mdoc_bd_pre(MDOC_ARGS);
71 static int       mdoc_bf_pre(MDOC_ARGS);
72 static void      mdoc_bk_post(MDOC_ARGS);
73 static int       mdoc_bk_pre(MDOC_ARGS);
74 static int       mdoc_bl_pre(MDOC_ARGS);
75 static int       mdoc_bt_pre(MDOC_ARGS);
76 static int       mdoc_bx_pre(MDOC_ARGS);
77 static int       mdoc_cd_pre(MDOC_ARGS);
78 static int       mdoc_dl_pre(MDOC_ARGS);
79 static int       mdoc_dv_pre(MDOC_ARGS);
80 static int       mdoc_fa_pre(MDOC_ARGS);
81 static int       mdoc_fd_pre(MDOC_ARGS);
82 static int       mdoc_fl_pre(MDOC_ARGS);
83 static int       mdoc_fn_pre(MDOC_ARGS);
84 static int       mdoc_ft_pre(MDOC_ARGS);
85 static int       mdoc_em_pre(MDOC_ARGS);
86 static int       mdoc_er_pre(MDOC_ARGS);
87 static int       mdoc_ev_pre(MDOC_ARGS);
88 static int       mdoc_ex_pre(MDOC_ARGS);
89 static void      mdoc_fo_post(MDOC_ARGS);
90 static int       mdoc_fo_pre(MDOC_ARGS);
91 static int       mdoc_ic_pre(MDOC_ARGS);
92 static int       mdoc_igndelim_pre(MDOC_ARGS);
93 static int       mdoc_in_pre(MDOC_ARGS);
94 static int       mdoc_it_pre(MDOC_ARGS);
95 static int       mdoc_lb_pre(MDOC_ARGS);
96 static int       mdoc_li_pre(MDOC_ARGS);
97 static int       mdoc_lk_pre(MDOC_ARGS);
98 static int       mdoc_mt_pre(MDOC_ARGS);
99 static int       mdoc_ms_pre(MDOC_ARGS);
100 static int       mdoc_nd_pre(MDOC_ARGS);
101 static int       mdoc_nm_pre(MDOC_ARGS);
102 static int       mdoc_ns_pre(MDOC_ARGS);
103 static int       mdoc_pa_pre(MDOC_ARGS);
104 static void      mdoc_pf_post(MDOC_ARGS);
105 static int       mdoc_pp_pre(MDOC_ARGS);
106 static void      mdoc_quote_post(MDOC_ARGS);
107 static int       mdoc_quote_pre(MDOC_ARGS);
108 static int       mdoc_rs_pre(MDOC_ARGS);
109 static int       mdoc_rv_pre(MDOC_ARGS);
110 static int       mdoc_sh_pre(MDOC_ARGS);
111 static int       mdoc_sm_pre(MDOC_ARGS);
112 static int       mdoc_sp_pre(MDOC_ARGS);
113 static int       mdoc_ss_pre(MDOC_ARGS);
114 static int       mdoc_sx_pre(MDOC_ARGS);
115 static int       mdoc_sy_pre(MDOC_ARGS);
116 static int       mdoc_ud_pre(MDOC_ARGS);
117 static int       mdoc_va_pre(MDOC_ARGS);
118 static int       mdoc_vt_pre(MDOC_ARGS);
119 static int       mdoc_xr_pre(MDOC_ARGS);
120 static int       mdoc_xx_pre(MDOC_ARGS);
121
122 static const struct htmlmdoc mdocs[MDOC_MAX] = {
```



```

123 {mdoc_ap_pre, NULL}, /* Ap */
124 {NULL, NULL}, /* Dd */
125 {NULL, NULL}, /* Dt */
126 {NULL, NULL}, /* Os */
127 {mdoc_sh_pre, NULL}, /* Sh */
128 {mdoc_ss_pre, NULL}, /* Ss */
129 {mdoc_pp_pre, NULL}, /* Pp */
130 {mdoc_dl_pre, NULL}, /* Dl */
131 {mdoc_dl_pre, NULL}, /* Dl */
132 {mdoc_bd_pre, NULL}, /* Bd */
133 {NULL, NULL}, /* Ed */
134 {mdoc_bl_pre, NULL}, /* Bl */
135 {NULL, NULL}, /* El */
136 {mdoc_it_pre, NULL}, /* It */
137 {mdoc_ad_pre, NULL}, /* Ad */
138 {mdoc_an_pre, NULL}, /* An */
139 {mdoc_ar_pre, NULL}, /* Ar */
140 {mdoc_cd_pre, NULL}, /* Cd */
141 {mdoc_fl_pre, NULL}, /* Cm */
142 {mdoc_dv_pre, NULL}, /* Dv */
143 {mdoc_er_pre, NULL}, /* Er */
144 {mdoc_ev_pre, NULL}, /* Ev */
145 {mdoc_ex_pre, NULL}, /* Ex */
146 {mdoc_fa_pre, NULL}, /* Fa */
147 {mdoc_fd_pre, NULL}, /* Fd */
148 {mdoc_fl_pre, NULL}, /* Fl */
149 {mdoc_fn_pre, NULL}, /* Fn */
150 {mdoc_ft_pre, NULL}, /* Ft */
151 {mdoc_ic_pre, NULL}, /* Ic */
152 {mdoc_in_pre, NULL}, /* In */
153 {mdoc_li_pre, NULL}, /* Li */
154 {mdoc_nd_pre, NULL}, /* Nd */
155 {mdoc_nm_pre, NULL}, /* Nm */
156 {mdoc_quote_pre, mdoc_quote_post}, /* Op */
157 {NULL, NULL}, /* Ot */
158 {mdoc_pa_pre, NULL}, /* Pa */
159 {mdoc_rv_pre, NULL}, /* Rv */
160 {NULL, NULL}, /* St */
161 {mdoc_va_pre, NULL}, /* Va */
162 {mdoc_vt_pre, NULL}, /* Vt */
163 {mdoc_xr_pre, NULL}, /* Xr */
164 {mdoc_x_pre, mdoc_x_post}, /* %A */
165 {mdoc_x_pre, mdoc_x_post}, /* %B */
166 {mdoc_x_pre, mdoc_x_post}, /* %D */
167 {mdoc_x_pre, mdoc_x_post}, /* %I */
168 {mdoc_x_pre, mdoc_x_post}, /* %J */
169 {mdoc_x_pre, mdoc_x_post}, /* %N */
170 {mdoc_x_pre, mdoc_x_post}, /* %O */
171 {mdoc_x_pre, mdoc_x_post}, /* %P */
172 {mdoc_x_pre, mdoc_x_post}, /* %R */
173 {mdoc_x_pre, mdoc_x_post}, /* %T */
174 {mdoc_x_pre, mdoc_x_post}, /* %V */
175 {NULL, NULL}, /* Ac */
176 {mdoc_quote_pre, mdoc_quote_post}, /* Ao */
177 {mdoc_quote_pre, mdoc_quote_post}, /* Aq */
178 {NULL, NULL}, /* At */
179 {NULL, NULL}, /* Bc */
180 {mdoc_bf_pre, NULL}, /* Bf */
181 {mdoc_quote_pre, mdoc_quote_post}, /* Bo */
182 {mdoc_quote_pre, mdoc_quote_post}, /* Bq */
183 {mdoc_xx_pre, NULL}, /* Bsx */
184 {mdoc_bx_pre, NULL}, /* Bx */
185 {NULL, NULL}, /* Db */
186 {NULL, NULL}, /* Dc */
187 {mdoc_quote_pre, mdoc_quote_post}, /* Do */
188 {mdoc_quote_pre, mdoc_quote_post}, /* Dq */

```

```

189 {NULL, NULL}, /* Ec */ /* FIXME: no space */
190 {NULL, NULL}, /* Ef */
191 {mdoc_em_pre, NULL}, /* Em */
192 {mdoc_quote_pre, mdoc_quote_post}, /* Eo */
193 {mdoc_xx_pre, NULL}, /* Fx */
194 {mdoc_ms_pre, NULL}, /* Ms */
195 {mdoc_igndelim_pre, NULL}, /* No */
196 {mdoc_ns_pre, NULL}, /* Ns */
197 {mdoc_xx_pre, NULL}, /* Nx */
198 {mdoc_xx_pre, NULL}, /* Ox */
199 {NULL, NULL}, /* Pc */
200 {mdoc_igndelim_pre, mdoc_pf_post}, /* Pf */
201 {mdoc_quote_pre, mdoc_quote_post}, /* Po */
202 {mdoc_quote_pre, mdoc_quote_post}, /* Pq */
203 {NULL, NULL}, /* Qc */
204 {mdoc_quote_pre, mdoc_quote_post}, /* Ql */
205 {mdoc_quote_pre, mdoc_quote_post}, /* Qo */
206 {mdoc_quote_pre, mdoc_quote_post}, /* Qq */
207 {NULL, NULL}, /* Re */
208 {mdoc_rs_pre, NULL}, /* Rs */
209 {NULL, NULL}, /* Sc */
210 {mdoc_quote_pre, mdoc_quote_post}, /* So */
211 {mdoc_quote_pre, mdoc_quote_post}, /* Sq */
212 {mdoc_sm_pre, NULL}, /* Sm */
213 {mdoc_sx_pre, NULL}, /* Sx */
214 {mdoc_sy_pre, NULL}, /* Sy */
215 {NULL, NULL}, /* Tn */
216 {mdoc_xx_pre, NULL}, /* Ux */
217 {NULL, NULL}, /* Xc */
218 {NULL, NULL}, /* Xo */
219 {mdoc_fo_pre, mdoc_fo_post}, /* Fo */
220 {NULL, NULL}, /* Fc */
221 {mdoc_quote_pre, mdoc_quote_post}, /* Oo */
222 {NULL, NULL}, /* Oc */
223 {mdoc_bk_pre, mdoc_bk_post}, /* Bk */
224 {NULL, NULL}, /* Ek */
225 {mdoc_bt_pre, NULL}, /* Bt */
226 {NULL, NULL}, /* Hf */
227 {NULL, NULL}, /* Fr */
228 {mdoc_ud_pre, NULL}, /* Ud */
229 {mdoc_lb_pre, NULL}, /* Lb */
230 {mdoc_pp_pre, NULL}, /* Lp */
231 {mdoc_lk_pre, NULL}, /* Lk */
232 {mdoc_mt_pre, NULL}, /* Mt */
233 {mdoc_quote_pre, mdoc_quote_post}, /* Brq */
234 {mdoc_quote_pre, mdoc_quote_post}, /* Bro */
235 {NULL, NULL}, /* Brc */
236 {mdoc_x_pre, mdoc_x_post}, /* %C */
237 {NULL, NULL}, /* Es */ /* TODO */
238 {NULL, NULL}, /* En */ /* TODO */
239 {mdoc_xx_pre, NULL}, /* Dx */
240 {mdoc_x_pre, mdoc_x_post}, /* %Q */
241 {mdoc_sp_pre, NULL}, /* br */
242 {mdoc_sp_pre, NULL}, /* sp */
243 {mdoc_x_pre, mdoc_x_post}, /* %U */
244 {NULL, NULL}, /* Ta */
245 };

247 static const char * const lists[LIST_MAX] = {
248     NULL,
249     "list-bul",
250     "list-col",
251     "list-dash",
252     "list-diag",
253     "list-enum",
254     "list-hang",

```

```

255     "list-hyph",
256     "list-inset",
257     "list-item",
258     "list-chang",
259     "list-tag"
260 };

262 void
263 html_mdock(void *arg, const struct mdock *m)
264 {

266     print_mdock(mdock_meta(m), mdock_node(m), (struct html *)arg);
267     putchar('\n');
268 }

271 /*
272  * Calculate the scaling unit passed in a '-width' argument. This uses
273  * either a native scaling unit (e.g., li, 2m) or the string length of
274  * the value.
275  */
276 static void
277 a2width(const char *p, struct roffsu *su)
278 {

280     if (! a2roffsu(p, su, SCALE_MAX)) {
281         su->unit = SCALE_BU;
282         su->scale = html_strlen(p);
283     }
284 }

287 /*
288  * See the same function in mdock_term.c for documentation.
289  */
290 static void
291 synopsis_pre(struct html *h, const struct mdock_node *n)
292 {

294     if (NULL == n->prev || ! (MDOCK_SYNPRETTY & n->flags))
295         return;

297     if (n->prev->tok == n->tok &&
298         MDOCK_Fo != n->tok &&
299         MDOCK_Ft != n->tok &&
300         MDOCK_Fn != n->tok) {
301         print_otag(h, TAG_BR, 0, NULL);
302         return;
303     }

305     switch (n->prev->tok) {
306     case (MDOCK_Fd):
307         /* FALLTHROUGH */
308     case (MDOCK_Fn):
309         /* FALLTHROUGH */
310     case (MDOCK_Fo):
311         /* FALLTHROUGH */
312     case (MDOCK_In):
313         /* FALLTHROUGH */
314     case (MDOCK_Vt):
315         print_otag(h, TAG_P, 0, NULL);
316         break;
317     case (MDOCK_Ft):
318         if (MDOCK_Fn != n->tok && MDOCK_Fo != n->tok) {
319             print_otag(h, TAG_P, 0, NULL);
320             break;

```

```

321     }
322     /* FALLTHROUGH */
323     default:
324         print_otag(h, TAG_BR, 0, NULL);
325         break;
326     }
327 }

330 /*
331  * Calculate the scaling unit passed in an '-offset' argument. This
332  * uses either a native scaling unit (e.g., li, 2m), one of a set of
333  * predefined strings (indent, etc.), or the string length of the value.
334  */
335 static void
336 a2offs(const char *p, struct roffsu *su)
337 {

339     /* FIXME: "right"? */

341     if (0 == strcmp(p, "left"))
342         SCALE_HS_INIT(su, 0);
343     else if (0 == strcmp(p, "indent"))
344         SCALE_HS_INIT(su, INDENT);
345     else if (0 == strcmp(p, "indent-two"))
346         SCALE_HS_INIT(su, INDENT * 2);
347     else if (! a2roffsu(p, su, SCALE_MAX))
348         SCALE_HS_INIT(su, html_strlen(p));
349 }

352 static void
353 print_mdock(MDOCK_ARGS)
354 {
355     struct tag *t, *tt;
356     struct htmlpair tag;

358     PAIR_CLASS_INIT(&tag, "mandock");

360     if (! (HTML_FRAGMENT & h->oflags)) {
361         print_gen_decls(h);
362         t = print_otag(h, TAG_HTML, 0, NULL);
363         tt = print_otag(h, TAG_HEAD, 0, NULL);
364         print_mdock_head(m, n, h);
365         print_tagq(h, tt);
366         print_otag(h, TAG_BODY, 0, NULL);
367         print_otag(h, TAG_DIV, 1, &tag);
368     } else
369         t = print_otag(h, TAG_DIV, 1, &tag);

371     print_mdock_nodelist(m, n, h);
372     print_tagq(h, t);
373 }

376 /* ARGSUSED */
377 static void
378 print_mdock_head(MDOCK_ARGS)
379 {

381     print_gen_head(h);
382     bufinit(h);
383     bufcat_fmt(h, "%s(%s)", m->title, m->msec);

385     if (m->arch)
386         bufcat_fmt(h, " (%s)", m->arch);

```

```

388     print_otag(h, TAG_TITLE, 0, NULL);
389     print_text(h, h->buf);
390 }

393 static void
394 print_mdoc_nodelist(MDOC_ARGS)
395 {
397     print_mdoc_node(m, n, h);
398     if (n->next)
399         print_mdoc_nodelist(m, n->next, h);
400 }

403 static void
404 print_mdoc_node(MDOC_ARGS)
405 {
406     int         child;
407     struct tag   *t;

409     child = 1;
410     t = h->tags.head;

412     switch (n->type) {
413     case (MDOC_ROOT):
414         child = mdoc_root_pre(m, n, h);
415         break;
416     case (MDOC_TEXT):
417         /* No tables in this mode... */
418         assert(NULL == h->tblt);

420         /*
421          * Make sure that if we're in a literal mode already
422          * (i.e., within a <PRE>) don't print the newline.
423          */
424         if (' ' == *n->string && MDOC_LINE & n->flags)
425             if (! (HTML_LITERAL & h->flags))
426                 print_otag(h, TAG_BR, 0, NULL);
427         if (MDOC_DELIMC & n->flags)
428             h->flags |= HTML_NOSPACE;
429         print_text(h, n->string);
430         if (MDOC_DELIMO & n->flags)
431             h->flags |= HTML_NOSPACE;
432         return;
433     case (MDOC_EQN):
434         print_eqn(h, n->eqn);
435         break;
436     case (MDOC_TBL):
437         /*
438          * This will take care of initialising all of the table
439          * state data for the first table, then tearing it down
440          * for the last one.
441          */
442         print_tbl(h, n->span);
443         return;
444     default:
445         /*
446          * Close out the current table, if it's open, and unset
447          * the "meta" table state. This will be reopened on the
448          * next table element.
449          */
450         if (h->tblt) {
451             print_tblclose(h);
452             t = h->tags.head;

```

```

453     }

455     assert(NULL == h->tblt);
456     if (mdocs[n->tok].pre && ENDBODY_NOT == n->end)
457         child = (*mdocs[n->tok].pre)(m, n, h);
458     break;
459 }

461     if (HTML_KEEP & h->flags) {
462         if (n->prev && n->prev->line != n->line) {
463             h->flags &= ~HTML_KEEP;
464             h->flags |= HTML_PREKEEP;
465         } else if (NULL == n->prev) {
466             if (n->parent && n->parent->line != n->line) {
467                 h->flags &= ~HTML_KEEP;
468                 h->flags |= HTML_PREKEEP;
469             }
470         }
471     }

473     if (child && n->child)
474         print_mdoc_nodelist(m, n->child, h);

476     print_stagq(h, t);

478     switch (n->type) {
479     case (MDOC_ROOT):
480         mdoc_root_post(m, n, h);
481         break;
482     case (MDOC_EQN):
483         break;
484     default:
485         if (mdocs[n->tok].post && ENDBODY_NOT == n->end)
486             (*mdocs[n->tok].post)(m, n, h);
487         break;
488     }
489 }

491 /* ARGSUSED */
492 static void
493 mdoc_root_post(MDOC_ARGS)
494 {
495     struct htmlpair tag[3];
496     struct tag       *t, *tt;

498     PAIR_SUMMARY_INIT(&tag[0], "Document Footer");
499     PAIR_CLASS_INIT(&tag[1], "foot");
500     PAIR_INIT(&tag[2], ATTR_WIDTH, "100%");
501     t = print_otag(h, TAG_TABLE, 3, tag);
502     PAIR_INIT(&tag[0], ATTR_WIDTH, "50%");
503     print_otag(h, TAG_COL, 1, tag);
504     print_otag(h, TAG_COL, 1, tag);

506     print_otag(h, TAG_TBODY, 0, NULL);

508     tt = print_otag(h, TAG_TR, 0, NULL);

510     PAIR_CLASS_INIT(&tag[0], "foot-date");
511     print_otag(h, TAG_TD, 1, tag);
512     print_text(h, m->date);
513     print_stagq(h, tt);

515     PAIR_CLASS_INIT(&tag[0], "foot-os");
516     PAIR_INIT(&tag[1], ATTR_ALIGN, "right");
517     print_otag(h, TAG_TD, 2, tag);
518     print_text(h, m->os);

```

```

519     print_tagq(h, t);
520 }

523 /* ARGSUSED */
524 static int
525 mdoc_root_pre(MDOC_ARGS)
526 {
527     struct htmlpair  tag[3];
528     struct tag       *t, *tt;
529     char              b[BUFSIZ], title[BUFSIZ];

531     strcpy(b, m->vol, BUFSIZ);

533     if (m->arch) {
534         strlcat(b, " (", BUFSIZ);
535         strlcat(b, m->arch, BUFSIZ);
536         strlcat(b, ")", BUFSIZ);
537     }

539     snprintf(title, BUFSIZ - 1, "%s(%s)", m->title, m->msec);

541     PAIR_SUMMARY_INIT(&tag[0], "Document Header");
542     PAIR_CLASS_INIT(&tag[1], "head");
543     PAIR_INIT(&tag[2], ATTR_WIDTH, "100%");
544     t = print_otag(h, TAG_TABLE, 3, tag);
545     PAIR_INIT(&tag[0], ATTR_WIDTH, "30%");
546     print_otag(h, TAG_COL, 1, tag);
547     print_otag(h, TAG_COL, 1, tag);
548     print_otag(h, TAG_COL, 1, tag);

550     print_otag(h, TAG_TBODY, 0, NULL);

552     tt = print_otag(h, TAG_TR, 0, NULL);

554     PAIR_CLASS_INIT(&tag[0], "head-ltitle");
555     print_otag(h, TAG_TD, 1, tag);
556     print_text(h, title);
557     print_stagq(h, tt);

559     PAIR_CLASS_INIT(&tag[0], "head-vol");
560     PAIR_INIT(&tag[1], ATTR_ALIGN, "center");
561     print_otag(h, TAG_TD, 2, tag);
562     print_text(h, b);
563     print_stagq(h, tt);

565     PAIR_CLASS_INIT(&tag[0], "head-rtitle");
566     PAIR_INIT(&tag[1], ATTR_ALIGN, "right");
567     print_otag(h, TAG_TD, 2, tag);
568     print_text(h, title);
569     print_tagq(h, t);
570     return(1);
571 }

574 /* ARGSUSED */
575 static int
576 mdoc_sh_pre(MDOC_ARGS)
577 {
578     struct htmlpair  tag;

580     if (MDOC_BLOCK == n->type) {
581         PAIR_CLASS_INIT(&tag, "section");
582         print_otag(h, TAG_DIV, 1, &tag);
583         return(1);
584     } else if (MDOC_BODY == n->type)

```

```

585         return(1);

587         bufinit(h);
588         bufcat(h, "x");

590         for (n = n->child; n && MDOC_TEXT == n->type; ) {
591             bufcat_id(h, n->string);
592             if (NULL != (n = n->next))
593                 bufcat_id(h, " ");
594         }

596         if (NULL == n) {
597             PAIR_ID_INIT(&tag, h->buf);
598             print_otag(h, TAG_H1, 1, &tag);
599         } else
600             print_otag(h, TAG_H1, 0, NULL);

602         return(1);
603     }

605 /* ARGSUSED */
606 static int
607 mdoc_ss_pre(MDOC_ARGS)
608 {
609     struct htmlpair  tag;

611     if (MDOC_BLOCK == n->type) {
612         PAIR_CLASS_INIT(&tag, "subsection");
613         print_otag(h, TAG_DIV, 1, &tag);
614         return(1);
615     } else if (MDOC_BODY == n->type)
616         return(1);

618     bufinit(h);
619     bufcat(h, "x");

621     for (n = n->child; n && MDOC_TEXT == n->type; ) {
622         bufcat_id(h, n->string);
623         if (NULL != (n = n->next))
624             bufcat_id(h, " ");
625     }

627     if (NULL == n) {
628         PAIR_ID_INIT(&tag, h->buf);
629         print_otag(h, TAG_H2, 1, &tag);
630     } else
631         print_otag(h, TAG_H2, 0, NULL);

633     return(1);
634 }

637 /* ARGSUSED */
638 static int
639 mdoc_fl_pre(MDOC_ARGS)
640 {
641     struct htmlpair  tag;

643     PAIR_CLASS_INIT(&tag, "flag");
644     print_otag(h, TAG_B, 1, &tag);

646     /* 'Cm' has no leading hyphen. */

648     if (MDOC_Cm == n->tok)
649         return(1);

```

```

651     print_text(h, "\\-");
652
653     if (n->child)
654         h->flags |= HTML_NOSPACE;
655     else if (n->next && n->next->line == n->line)
656         h->flags |= HTML_NOSPACE;
657
658     return(1);
659 }
660
661 /* ARGSUSED */
662 static int
663 mdoc_nd_pre(MDOC_ARGS)
664 {
665     struct htmlpair tag;
666
667     if (MDOC_BODY != n->type)
668         return(1);
669
670     /* XXX: this tag in theory can contain block elements. */
671
672     print_text(h, "\\(em");
673     PAIR_CLASS_INIT(&tag, "desc");
674     print_otag(h, TAG_SPAN, 1, &tag);
675     return(1);
676 }
677
678 static int
679 mdoc_nm_pre(MDOC_ARGS)
680 {
681     struct htmlpair tag;
682     struct roffsu su;
683     int len;
684
685     switch (n->type) {
686     case (MDOC_ELEM):
687         synopsis_pre(h, n);
688         PAIR_CLASS_INIT(&tag, "name");
689         print_otag(h, TAG_B, 1, &tag);
690         if (NULL == n->child && m->name)
691             print_text(h, m->name);
692         return(1);
693     case (MDOC_HEAD):
694         print_otag(h, TAG_TD, 0, NULL);
695         if (NULL == n->child && m->name)
696             print_text(h, m->name);
697         return(1);
698     case (MDOC_BODY):
699         print_otag(h, TAG_TD, 0, NULL);
700         return(1);
701     default:
702         break;
703     }
704
705     synopsis_pre(h, n);
706     PAIR_CLASS_INIT(&tag, "synopsis");
707     print_otag(h, TAG_TABLE, 1, &tag);
708
709     for (len = 0, n = n->child; n; n = n->next)
710         if (MDOC_TEXT == n->type)
711             len += html_strlen(n->string);
712
713     if (0 == len && m->name)
714         len = html_strlen(m->name);

```

```

718     SCALE_HS_INIT(&su, (double)len);
719     bufinit(h);
720     bufcat_su(h, "width", &su);
721     PAIR_STYLE_INIT(&tag, h);
722     print_otag(h, TAG_COL, 1, &tag);
723     print_otag(h, TAG_COL, 0, NULL);
724     print_otag(h, TAG_TBODY, 0, NULL);
725     print_otag(h, TAG_TR, 0, NULL);
726     return(1);
727 }
728
729 /* ARGSUSED */
730 static int
731 mdoc_xr_pre(MDOC_ARGS)
732 {
733     struct htmlpair tag[2];
734
735     if (NULL == n->child)
736         return(0);
737
738     PAIR_CLASS_INIT(&tag[0], "link-man");
739
740     if (h->base_man) {
741         buffmt_man(h, n->child->string,
742                 n->child->next ?
743                 n->child->next->string : NULL);
744         PAIR_HREF_INIT(&tag[1], h->buf);
745         print_otag(h, TAG_A, 2, tag);
746     } else
747         print_otag(h, TAG_A, 1, tag);
748
749     n = n->child;
750     print_text(h, n->string);
751
752     if (NULL == (n = n->next))
753         return(0);
754
755     h->flags |= HTML_NOSPACE;
756     print_text(h, "(");
757     h->flags |= HTML_NOSPACE;
758     print_text(h, n->string);
759     h->flags |= HTML_NOSPACE;
760     print_text(h, ")");
761     return(0);
762 }
763
764 /* ARGSUSED */
765 static int
766 mdoc_ns_pre(MDOC_ARGS)
767 {
768     if ( ! (MDOC_LINE & n->flags))
769         h->flags |= HTML_NOSPACE;
770     return(1);
771 }
772
773 /* ARGSUSED */
774 static int
775 mdoc_ar_pre(MDOC_ARGS)
776 {
777     struct htmlpair tag;

```

```

783     PAIR_CLASS_INIT(&tag, "arg");
784     print_otag(h, TAG_I, 1, &tag);
785     return(1);
786 }

789 /* ARGSUSED */
790 static int
791 mdoc_xx_pre(MDOC_ARGS)
792 {
793     const char    *pp;
794     struct htmlpair tag;
795     int           flags;

797     switch (n->tok) {
798     case (MDOC_Bsx):
799         pp = "BSD/OS";
800         break;
801     case (MDOC_Dx):
802         pp = "DragonFly";
803         break;
804     case (MDOC_Fx):
805         pp = "FreeBSD";
806         break;
807     case (MDOC_Nx):
808         pp = "NetBSD";
809         break;
810     case (MDOC_Ox):
811         pp = "OpenBSD";
812         break;
813     case (MDOC_Ux):
814         pp = "UNIX";
815         break;
816     default:
817         return(1);
818     }

820     PAIR_CLASS_INIT(&tag, "unix");
821     print_otag(h, TAG_SPAN, 1, &tag);

823     print_text(h, pp);
824     if (n->child) {
825         flags = h->flags;
826         h->flags |= HTML_KEEP;
827         print_text(h, n->child->string);
828         h->flags = flags;
829     }
830     return(0);
831 }

834 /* ARGSUSED */
835 static int
836 mdoc_bx_pre(MDOC_ARGS)
837 {
838     struct htmlpair tag;

840     PAIR_CLASS_INIT(&tag, "unix");
841     print_otag(h, TAG_SPAN, 1, &tag);

843     if (NULL != (n = n->child)) {
844         print_text(h, n->string);
845         h->flags |= HTML_NOSPACE;
846         print_text(h, "BSD");
847     } else {
848         print_text(h, "BSD");

```

```

849         return(0);
850     }

852     if (NULL != (n = n->next)) {
853         h->flags |= HTML_NOSPACE;
854         print_text(h, "-");
855         h->flags |= HTML_NOSPACE;
856         print_text(h, n->string);
857     }

859     return(0);
860 }

862 /* ARGSUSED */
863 static int
864 mdoc_it_pre(MDOC_ARGS)
865 {
866     struct roffsu    su;
867     enum mdoc_list   type;
868     struct htmlpair  tag[2];
869     const struct mdoc_node *bl;

871     bl = n->parent;
872     while (bl && MDOC_Bl != bl->tok)
873         bl = bl->parent;

875     assert(bl);

877     type = bl->norm->Bl.type;

879     assert(lists[type]);
880     PAIR_CLASS_INIT(&tag[0], lists[type]);

882     bufinit(h);

884     if (MDOC_HEAD == n->type) {
885         switch (type) {
886         case (LIST_bullet):
887             /* FALLTHROUGH */
888         case (LIST_dash):
889             /* FALLTHROUGH */
890         case (LIST_item):
891             /* FALLTHROUGH */
892         case (LIST_hyphen):
893             /* FALLTHROUGH */
894         case (LIST_enum):
895             return(0);
896         case (LIST_diag):
897             /* FALLTHROUGH */
898         case (LIST_hang):
899             /* FALLTHROUGH */
900         case (LIST_inset):
901             /* FALLTHROUGH */
902         case (LIST_ohang):
903             /* FALLTHROUGH */
904         case (LIST_tag):
905             SCALE_VS_INIT(&su, ! bl->norm->Bl.comp);
906             bufcat_su(h, "margin-top", &su);
907             PAIR_STYLE_INIT(&tag[1], h);
908             print_otag(h, TAG_DT, 2, tag);
909             if (LIST_diag != type)
910                 break;
911             PAIR_CLASS_INIT(&tag[0], "diag");
912             print_otag(h, TAG_B, 1, tag);
913             break;
914         case (LIST_column):

```

```

915         break;
916     default:
917         break;
918     }
919 } else if (MDOC_BODY == n->type) {
920     switch (type) {
921     case(LIST_bullet):
922         /* FALLTHROUGH */
923     case(LIST_hyphen):
924         /* FALLTHROUGH */
925     case(LIST_dash):
926         /* FALLTHROUGH */
927     case(LIST_enum):
928         /* FALLTHROUGH */
929     case(LIST_item):
930         SCALE_VS_INIT(&su, ! bl->norm->Bl.comp);
931         bufcat_su(h, "margin-top", &su);
932         PAIR_STYLE_INIT(&tag[1], h);
933         print_otag(h, TAG_LI, 2, tag);
934         break;
935     case(LIST_diag):
936         /* FALLTHROUGH */
937     case(LIST_hang):
938         /* FALLTHROUGH */
939     case(LIST_inset):
940         /* FALLTHROUGH */
941     case(LIST_ohang):
942         /* FALLTHROUGH */
943     case(LIST_tag):
944         if (NULL == bl->norm->Bl.width) {
945             print_otag(h, TAG_DD, 1, tag);
946             break;
947         }
948         a2width(bl->norm->Bl.width, &su);
949         bufcat_su(h, "margin-left", &su);
950         PAIR_STYLE_INIT(&tag[1], h);
951         print_otag(h, TAG_DD, 2, tag);
952         break;
953     case(LIST_column):
954         SCALE_VS_INIT(&su, ! bl->norm->Bl.comp);
955         bufcat_su(h, "margin-top", &su);
956         PAIR_STYLE_INIT(&tag[1], h);
957         print_otag(h, TAG_TD, 2, tag);
958         break;
959     default:
960         break;
961     }
962 } else {
963     switch (type) {
964     case(LIST_column):
965         print_otag(h, TAG_TR, 1, tag);
966         break;
967     default:
968         break;
969     }
970 }
971
972 return(1);
973 }
974
975 /* ARGSUSED */
976 static int
977 mdoc_bl_pre(MDOC_ARGS)
978 {
979     int i;
980     struct htmlpair tag[3];

```

```

981     struct roffsu su;
982     char buf[BUFSIZ];
983
984     bufinit(h);
985
986     if (MDOC_BODY == n->type) {
987         if (LIST_column == n->norm->Bl.type)
988             print_otag(h, TAG_TBODY, 0, NULL);
989         return(1);
990     }
991
992     if (MDOC_HEAD == n->type) {
993         if (LIST_column != n->norm->Bl.type)
994             return(0);
995
996         /*
997          * For each column, print out the <COL> tag with our
998          * suggested width. The last column gets min-width, as
999          * in terminal mode it auto-sizes to the width of the
1000          * screen and we want to preserve that behaviour.
1001          */
1002
1003         for (i = 0; i < (int)n->norm->Bl.ncols; i++) {
1004             a2width(n->norm->Bl.cols[i], &su);
1005             if (i < (int)n->norm->Bl.ncols - 1)
1006                 bufcat_su(h, "width", &su);
1007             else
1008                 bufcat_su(h, "min-width", &su);
1009             PAIR_STYLE_INIT(&tag[0], h);
1010             print_otag(h, TAG_COL, 1, tag);
1011         }
1012
1013         return(0);
1014     }
1015
1016     SCALE_VS_INIT(&su, 0);
1017     bufcat_su(h, "margin-top", &su);
1018     bufcat_su(h, "margin-bottom", &su);
1019     PAIR_STYLE_INIT(&tag[0], h);
1020
1021     assert(lists[n->norm->Bl.type]);
1022     strcpy(buf, "list ", BUFSIZ);
1023     strcat(buf, lists[n->norm->Bl.type], BUFSIZ);
1024     PAIR_INIT(&tag[1], ATTR_CLASS, buf);
1025
1026     /* Set the block's left-hand margin. */
1027
1028     if (n->norm->Bl.off5) {
1029         a2offs(n->norm->Bl.off5, &su);
1030         bufcat_su(h, "margin-left", &su);
1031     }
1032
1033     switch (n->norm->Bl.type) {
1034     case(LIST_bullet):
1035         /* FALLTHROUGH */
1036     case(LIST_dash):
1037         /* FALLTHROUGH */
1038     case(LIST_hyphen):
1039         /* FALLTHROUGH */
1040     case(LIST_item):
1041         print_otag(h, TAG_UL, 2, tag);
1042         break;
1043     case(LIST_enum):
1044         print_otag(h, TAG_OL, 2, tag);
1045         break;
1046     case(LIST_diag):

```

```

1047         /* FALLTHROUGH */
1048     case(LIST_hang):
1049         /* FALLTHROUGH */
1050     case(LIST_inset):
1051         /* FALLTHROUGH */
1052     case(LIST_ohang):
1053         /* FALLTHROUGH */
1054     case(LIST_tag):
1055         print_otag(h, TAG_DL, 2, tag);
1056         break;
1057     case(LIST_column):
1058         print_otag(h, TAG_TABLE, 2, tag);
1059         break;
1060     default:
1061         abort();
1062         /* NOTREACHED */
1063     }
1064
1065     return(1);
1066 }
1067
1068 /* ARGSUSED */
1069 static int
1070 mdoc_ex_pre(MDOC_ARGS)
1071 {
1072     struct tag      *t;
1073     struct htmlpair tag;
1074     int             nchild;
1075
1076     if (n->prev)
1077         print_otag(h, TAG_BR, 0, NULL);
1078
1079     PAIR_CLASS_INIT(&tag, "utility");
1080
1081     print_text(h, "The");
1082
1083     nchild = n->nchild;
1084     for (n = n->child; n; n = n->next) {
1085         assert(MDOC_TEXT == n->type);
1086
1087         t = print_otag(h, TAG_B, 1, &tag);
1088         print_text(h, n->string);
1089         print_tagq(h, t);
1090
1091         if (nchild > 2 && n->next) {
1092             h->flags |= HTML_NOSPACE;
1093             print_text(h, ",");
1094         }
1095
1096         if (n->next && NULL == n->next->next)
1097             print_text(h, "and");
1098     }
1099
1100     if (nchild > 1)
1101         print_text(h, "utilities exit");
1102     else
1103         print_text(h, "utility exits");
1104
1105     print_text(h, "0 on success, and >0 if an error occurs.");
1106     return(0);
1107 }
1108
1109 /* ARGSUSED */
1110 static int
1111 mdoc_em_pre(MDOC_ARGS)

```

```

1112 {
1113     struct htmlpair tag;
1114
1115     PAIR_CLASS_INIT(&tag, "emph");
1116     print_otag(h, TAG_SPAN, 1, &tag);
1117     return(1);
1118 }
1119
1120 /* ARGSUSED */
1121 static int
1122 mdoc_dl_pre(MDOC_ARGS)
1123 {
1124     struct htmlpair tag[2];
1125     struct roffsu    su;
1126
1127     if (MDOC_BLOCK != n->type)
1128         return(1);
1129
1130     SCALE_VS_INIT(&su, 0);
1131     bufinit(h);
1132     bufcat_su(h, "margin-top", &su);
1133     bufcat_su(h, "margin-bottom", &su);
1134     PAIR_STYLE_INIT(&tag[0], h);
1135     print_otag(h, TAG_BLOCKQUOTE, 1, tag);
1136
1137     /* BLOCKQUOTE needs a block body. */
1138
1139     PAIR_CLASS_INIT(&tag[0], "display");
1140     print_otag(h, TAG_DIV, 1, tag);
1141
1142     if (MDOC_Dl == n->tok) {
1143         PAIR_CLASS_INIT(&tag[0], "lit");
1144         print_otag(h, TAG_CODE, 1, tag);
1145     }
1146
1147     return(1);
1148 }
1149
1150 /* ARGSUSED */
1151 static int
1152 mdoc_sx_pre(MDOC_ARGS)
1153 {
1154     struct htmlpair tag[2];
1155
1156     bufinit(h);
1157     bufcat(h, "#x");
1158
1159     for (n = n->child; n; ) {
1160         bufcat_id(h, n->string);
1161         if (NULL != (n = n->next))
1162             bufcat_id(h, " ");
1163     }
1164
1165     PAIR_CLASS_INIT(&tag[0], "link-sec");
1166     PAIR_HREF_INIT(&tag[1], h->buf);
1167
1168     print_otag(h, TAG_I, 1, tag);
1169     print_otag(h, TAG_A, 2, tag);
1170     return(1);
1171 }
1172
1173 /* ARGSUSED */
1174 static int

```



```

1179 mdoc_bd_pre(MDOC_ARGS)
1180 {
1181     struct htmlpair      tag[2];
1182     int                  comp, sv;
1183     const struct mdoc_node *nn;
1184     struct roffsu        su;
1185
1186     if (MDOC_HEAD == n->type)
1187         return(0);
1188
1189     if (MDOC_BLOCK == n->type) {
1190         comp = n->norm->Bd.comp;
1191         for (nn = n; nn && ! comp; nn = nn->parent) {
1192             if (MDOC_BLOCK != nn->type)
1193                 continue;
1194             if (MDOC_Ss == nn->tok || MDOC_Sh == nn->tok)
1195                 comp = 1;
1196             if (nn->prev)
1197                 break;
1198         }
1199         if (! comp)
1200             print_otag(h, TAG_P, 0, NULL);
1201         return(1);
1202     }
1203
1204     SCALE_HS_INIT(&su, 0);
1205     if (n->norm->Bd.offfs)
1206         a2offs(n->norm->Bd.offfs, &su);
1207
1208     bufinit(h);
1209     bufcat_su(h, "margin-left", &su);
1210     PAIR_STYLE_INIT(&tag[0], h);
1211
1212     if (DISP_unfilled != n->norm->Bd.type &&
1213         DISP_literal != n->norm->Bd.type) {
1214         PAIR_CLASS_INIT(&tag[1], "display");
1215         print_otag(h, TAG_DIV, 2, tag);
1216         return(1);
1217     }
1218
1219     PAIR_CLASS_INIT(&tag[1], "lit display");
1220     print_otag(h, TAG_PRE, 2, tag);
1221
1222     /* This can be recursive: save & set our literal state. */
1223
1224     sv = h->flags & HTML_LITERAL;
1225     h->flags |= HTML_LITERAL;
1226
1227     for (nn = n->child; nn; nn = nn->next) {
1228         print_mdoc_node(m, nn, h);
1229         /*
1230          * If the printed node flushes its own line, then we
1231          * needn't do it here as well. This is hacky, but the
1232          * notion of selective eoln whitespace is pretty dumb
1233          * anyway, so don't sweat it.
1234          */
1235         switch (nn->tok) {
1236             case (MDOC_Sm):
1237                 /* FALLTHROUGH */
1238             case (MDOC_br):
1239                 /* FALLTHROUGH */
1240             case (MDOC_sp):
1241                 /* FALLTHROUGH */
1242             case (MDOC_Bl):
1243                 /* FALLTHROUGH */
1244             case (MDOC_Dl):

```

```

1245         /* FALLTHROUGH */
1246         case (MDOC_Dl):
1247             /* FALLTHROUGH */
1248         case (MDOC_Lp):
1249             /* FALLTHROUGH */
1250         case (MDOC_Pp):
1251             continue;
1252         default:
1253             break;
1254     }
1255     if (nn->next && nn->next->line == nn->line)
1256         continue;
1257     else if (nn->next)
1258         print_text(h, "\n");
1259
1260     h->flags |= HTML_NOSPACE;
1261 }
1262
1263 if (0 == sv)
1264     h->flags &= ~HTML_LITERAL;
1265
1266 return(0);
1267 }
1268
1269 /* ARGSUSED */
1270 static int
1271 mdoc_pa_pre(MDOC_ARGS)
1272 {
1273     struct htmlpair tag;
1274
1275     PAIR_CLASS_INIT(&tag, "file");
1276     print_otag(h, TAG_I, 1, &tag);
1277     return(1);
1278 }
1279
1280 /* ARGSUSED */
1281 static int
1282 mdoc_ad_pre(MDOC_ARGS)
1283 {
1284     struct htmlpair tag;
1285
1286     PAIR_CLASS_INIT(&tag, "addr");
1287     print_otag(h, TAG_I, 1, &tag);
1288     return(1);
1289 }
1290
1291 /* ARGSUSED */
1292 static int
1293 mdoc_an_pre(MDOC_ARGS)
1294 {
1295     struct htmlpair tag;
1296
1297     /* TODO: -split and -nosplit (see term_p_an_pre()). */
1298
1299     PAIR_CLASS_INIT(&tag, "author");
1300     print_otag(h, TAG_SPAN, 1, &tag);
1301     return(1);
1302 }
1303
1304 /* ARGSUSED */
1305 static int
1306 mdoc_cd_pre(MDOC_ARGS)

```

```

1311 {
1312     struct htmlpair tag;

1314     synopsis_pre(h, n);
1315     PAIR_CLASS_INIT(&tag, "config");
1316     print_otag(h, TAG_B, 1, &tag);
1317     return(1);
1318 }

1321 /* ARGSUSED */
1322 static int
1323 mdoc_dv_pre(MDOC_ARGS)
1324 {
1325     struct htmlpair tag;

1327     PAIR_CLASS_INIT(&tag, "define");
1328     print_otag(h, TAG_SPAN, 1, &tag);
1329     return(1);
1330 }

1333 /* ARGSUSED */
1334 static int
1335 mdoc_ev_pre(MDOC_ARGS)
1336 {
1337     struct htmlpair tag;

1339     PAIR_CLASS_INIT(&tag, "env");
1340     print_otag(h, TAG_SPAN, 1, &tag);
1341     return(1);
1342 }

1345 /* ARGSUSED */
1346 static int
1347 mdoc_er_pre(MDOC_ARGS)
1348 {
1349     struct htmlpair tag;

1351     PAIR_CLASS_INIT(&tag, "errno");
1352     print_otag(h, TAG_SPAN, 1, &tag);
1353     return(1);
1354 }

1357 /* ARGSUSED */
1358 static int
1359 mdoc_fa_pre(MDOC_ARGS)
1360 {
1361     const struct mdoc_node *nn;
1362     struct htmlpair tag;
1363     struct tag *t;

1365     PAIR_CLASS_INIT(&tag, "farg");
1366     if (n->parent->tok != MDOC_Fo) {
1367         print_otag(h, TAG_I, 1, &tag);
1368         return(1);
1369     }

1371     for (nn = n->child; nn; nn = nn->next) {
1372         t = print_otag(h, TAG_I, 1, &tag);
1373         print_text(h, nn->string);
1374         print_tagq(h, t);
1375         if (nn->next) {
1376             h->flags |= HTML_NOSPACE;

```

```

1377         print_text(h, ",");
1378     }
1379 }

1381     if (n->child && n->next && n->next->tok == MDOC_Fa) {
1382         h->flags |= HTML_NOSPACE;
1383         print_text(h, ",");
1384     }

1386     return(0);
1387 }

1390 /* ARGSUSED */
1391 static int
1392 mdoc_fd_pre(MDOC_ARGS)
1393 {
1394     struct htmlpair tag[2];
1395     char buf[BUFSIZ];
1396     size_t sz;
1397     int i;
1398     struct tag *t;

1400     synopsis_pre(h, n);

1402     if (NULL == (n = n->child))
1403         return(0);

1405     assert(MDOC_TEXT == n->type);

1407     if (strcmp(n->string, "#include")) {
1408         PAIR_CLASS_INIT(&tag[0], "macro");
1409         print_otag(h, TAG_B, 1, tag);
1410         return(1);
1411     }

1413     PAIR_CLASS_INIT(&tag[0], "includes");
1414     print_otag(h, TAG_B, 1, tag);
1415     print_text(h, n->string);

1417     if (NULL != (n = n->next)) {
1418         assert(MDOC_TEXT == n->type);
1419         strcpy(buf, '<' == *n->string || '"' == *n->string ?
1420             n->string + 1 : n->string, BUFSIZ);

1422         sz = strlen(buf);
1423         if (sz && ('>' == buf[sz - 1] || '"' == buf[sz - 1]))
1424             buf[sz - 1] = '\0';

1426         PAIR_CLASS_INIT(&tag[0], "link-includes");
1427
1428         i = 1;
1429         if (h->base_includes) {
1430             buffmt_includes(h, buf);
1431             PAIR_HREF_INIT(&tag[i], h->buf);
1432             i++;
1433         }

1435         t = print_otag(h, TAG_A, i, tag);
1436         print_text(h, n->string);
1437         print_tagq(h, t);

1439         n = n->next;
1440     }

1442     for ( ; n; n = n->next) {

```

```

1443         assert(MDOC_TEXT == n->type);
1444         print_text(h, n->string);
1445     }
1447     return(0);
1448 }

1451 /* ARGSUSED */
1452 static int
1453 mdoc_vt_pre(MDOC_ARGS)
1454 {
1455     struct htmlpair tag;

1457     if (MDOC_BLOCK == n->type) {
1458         synopsis_pre(h, n);
1459         return(1);
1460     } else if (MDOC_ELEM == n->type) {
1461         synopsis_pre(h, n);
1462     } else if (MDOC_HEAD == n->type)
1463         return(0);

1465     PAIR_CLASS_INIT(&tag, "type");
1466     print_otag(h, TAG_SPAN, 1, &tag);
1467     return(1);
1468 }

1471 /* ARGSUSED */
1472 static int
1473 mdoc_ft_pre(MDOC_ARGS)
1474 {
1475     struct htmlpair tag;

1477     synopsis_pre(h, n);
1478     PAIR_CLASS_INIT(&tag, "ftype");
1479     print_otag(h, TAG_I, 1, &tag);
1480     return(1);
1481 }

1484 /* ARGSUSED */
1485 static int
1486 mdoc_fn_pre(MDOC_ARGS)
1487 {
1488     struct tag *t;
1489     struct htmlpair tag[2];
1490     char nbuf[BUFSIZ];
1491     const char *sp, *ep;
1492     int sz, i, pretty;

1494     pretty = MDOC_SYNPRETTY & n->flags;
1495     synopsis_pre(h, n);

1497     /* Split apart into type and name. */
1498     assert(n->child->string);
1499     sp = n->child->string;

1501     ep = strchr(sp, ' ');
1502     if (NULL != ep) {
1503         PAIR_CLASS_INIT(&tag[0], "ftype");
1504         t = print_otag(h, TAG_I, 1, tag);
1505     }
1506     while (ep) {
1507         sz = MIN((int)(ep - sp), BUFSIZ - 1);
1508         (void)memcpy(nbuf, sp, (size_t)sz);

```

```

1509         nbuf[sz] = '\0';
1510         print_text(h, nbuf);
1511         sp = ++ep;
1512         ep = strchr(sp, ' ');
1513     }
1514     print_tagq(h, t);
1515 }

1517 PAIR_CLASS_INIT(&tag[0], "fname");

1519 /*
1520  * FIXME: only refer to IDs that we know exist.
1521  */

1523 #if 0
1524     if (MDOC_SYNPRETTY & n->flags) {
1525         nbuf[0] = '\0';
1526         html_idcat(nbuf, sp, BUFSIZ);
1527         PAIR_ID_INIT(&tag[1], nbuf);
1528     } else {
1529         strlcpy(nbuf, "#", BUFSIZ);
1530         html_idcat(nbuf, sp, BUFSIZ);
1531         PAIR_HREF_INIT(&tag[1], nbuf);
1532     }
1533 #endif

1535     t = print_otag(h, TAG_B, 1, tag);

1537     if (sp) {
1538         strlcpy(nbuf, sp, BUFSIZ);
1539         print_text(h, nbuf);
1540     }

1542     print_tagq(h, t);

1544     h->flags |= HTML_NOSPACE;
1545     print_text(h, "");
1546     h->flags |= HTML_NOSPACE;

1548     PAIR_CLASS_INIT(&tag[0], "farg");
1549     bufinit(h);
1550     bufcat_style(h, "white-space", "nowrap");
1551     PAIR_STYLE_INIT(&tag[1], h);

1553     for (n = n->child->next; n; n = n->next) {
1554         i = 1;
1555         if (MDOC_SYNPRETTY & n->flags)
1556             i = 2;
1557         t = print_otag(h, TAG_I, i, tag);
1558         print_text(h, n->string);
1559         print_tagq(h, t);
1560         if (n->next) {
1561             h->flags |= HTML_NOSPACE;
1562             print_text(h, "");
1563         }
1564     }

1566     h->flags |= HTML_NOSPACE;
1567     print_text(h, "");

1569     if (pretty) {
1570         h->flags |= HTML_NOSPACE;
1571         print_text(h, "");
1572     }

1574     return(0);

```

```

1575 }

1578 /* ARGSUSED */
1579 static int
1580 mdoc_sm_pre(MDOC_ARGS)
1581 {
1583     assert(n->child && MDOC_TEXT == n->child->type);
1584     if (0 == strcmp("on", n->child->string)) {
1585         /*
1586          * FIXME: no p->col to check.  Thus, if we have
1587          * .Bd -literal
1588          * .Sm off
1589          * 1 2
1590          * .Sm on
1591          * 3
1592          * .Ed
1593          * the "3" is preceded by a space.
1594          */
1595         h->flags &= ~HTML_NOSPACE;
1596         h->flags &= ~HTML_NONOSPACE;
1597     } else
1598         h->flags |= HTML_NONOSPACE;

1600     return(0);
1601 }

1603 /* ARGSUSED */
1604 static int
1605 mdoc_pp_pre(MDOC_ARGS)
1606 {
1608     print_otag(h, TAG_P, 0, NULL);
1609     return(0);
1611 }

1613 /* ARGSUSED */
1614 static int
1615 mdoc_sp_pre(MDOC_ARGS)
1616 {
1617     struct roffsu    su;
1618     struct htmlpair tag;

1620     SCALE_VS_INIT(&su, 1);

1622     if (MDOC_sp == n->tok) {
1623         if (NULL != (n = n->child))
1624             if (! a2roffsu(n->string, &su, SCALE_VS))
1625                 SCALE_VS_INIT(&su, atoi(n->string));
1626     } else
1627         su.scale = 0;

1629     bufinit(h);
1630     bufcat_su(h, "height", &su);
1631     PAIR_STYLE_INIT(&tag, h);
1632     print_otag(h, TAG_DIV, 1, &tag);

1634     /* So the div isn't empty: */
1635     print_text(h, "\\~");

1637     return(0);

1639 }

```

```

1641 /* ARGSUSED */
1642 static int
1643 mdoc_lk_pre(MDOC_ARGS)
1644 {
1645     struct htmlpair tag[2];

1647     if (NULL == (n = n->child))
1648         return(0);

1650     assert(MDOC_TEXT == n->type);

1652     PAIR_CLASS_INIT(&tag[0], "link-ext");
1653     PAIR_HREF_INIT(&tag[1], n->string);

1655     print_otag(h, TAG_A, 2, tag);

1657     if (NULL == n->next)
1658         print_text(h, n->string);

1660     for (n = n->next; n; n = n->next)
1661         print_text(h, n->string);

1663     return(0);
1664 }

1667 /* ARGSUSED */
1668 static int
1669 mdoc_mt_pre(MDOC_ARGS)
1670 {
1671     struct htmlpair tag[2];
1672     struct tag      *t;

1674     PAIR_CLASS_INIT(&tag[0], "link-mail");

1676     for (n = n->child; n; n = n->next) {
1677         assert(MDOC_TEXT == n->type);

1679         bufinit(h);
1680         bufcat(h, "mailto:");
1681         bufcat(h, n->string);

1683         PAIR_HREF_INIT(&tag[1], h->buf);
1684         t = print_otag(h, TAG_A, 2, tag);
1685         print_text(h, n->string);
1686         print_tagq(h, t);
1687     }

1688     return(0);
1689 }
1690 }

1693 /* ARGSUSED */
1694 static int
1695 mdoc_fo_pre(MDOC_ARGS)
1696 {
1697     struct htmlpair tag;
1698     struct tag      *t;

1700     if (MDOC_BODY == n->type) {
1701         h->flags |= HTML_NOSPACE;
1702         print_text(h, "(");
1703         h->flags |= HTML_NOSPACE;
1704         return(1);
1705     } else if (MDOC_BLOCK == n->type) {
1706         synopsis_pre(h, n);

```

```

1707         return(1);
1708     }

1710     /* XXX: we drop non-initial arguments as per groff. */

1712     assert(n->child);
1713     assert(n->child->string);

1715     PAIR_CLASS_INIT(&tag, "fname");
1716     t = print_otag(h, TAG_B, 1, &tag);
1717     print_text(h, n->child->string);
1718     print_tagq(h, t);
1719     return(0);
1720 }

1723 /* ARGSUSED */
1724 static void
1725 mdoc_fo_post(MDOC_ARGS)
1726 {

1728     if (MDOC_BODY != n->type)
1729         return;
1730     h->flags |= HTML_NOSPACE;
1731     print_text(h, "");
1732     h->flags |= HTML_NOSPACE;
1733     print_text(h, ";");
1734 }

1737 /* ARGSUSED */
1738 static int
1739 mdoc_in_pre(MDOC_ARGS)
1740 {
1741     struct tag      *t;
1742     struct htmlpair tag[2];
1743     int             i;

1745     synopsis_pre(h, n);

1747     PAIR_CLASS_INIT(&tag[0], "includes");
1748     print_otag(h, TAG_B, 1, tag);

1750     /*
1751     * The first argument of the 'In' gets special treatment as
1752     * being a linked value. Subsequent values are printed
1753     * afterward. groff does similarly. This also handles the case
1754     * of no children.
1755     */

1757     if (MDOC_SYNPRETTY & n->flags && MDOC_LINE & n->flags)
1758         print_text(h, "#include");

1760     print_text(h, "<");
1761     h->flags |= HTML_NOSPACE;

1763     if (NULL != (n = n->child)) {
1764         assert(MDOC_TEXT == n->type);

1766         PAIR_CLASS_INIT(&tag[0], "link-includes");

1768         i = 1;
1769         if (h->base_includes) {
1770             buffmt_includes(h, n->string);
1771             PAIR_HREF_INIT(&tag[i], h->buf);
1772             i++;

```

```

1773     }

1775     t = print_otag(h, TAG_A, i, tag);
1776     print_text(h, n->string);
1777     print_tagq(h, t);

1779     n = n->next;
1780 }

1782     h->flags |= HTML_NOSPACE;
1783     print_text(h, ">");

1785     for ( ; n; n = n->next) {
1786         assert(MDOC_TEXT == n->type);
1787         print_text(h, n->string);
1788     }

1790     return(0);
1791 }

1794 /* ARGSUSED */
1795 static int
1796 mdoc_ic_pre(MDOC_ARGS)
1797 {
1798     struct htmlpair tag;

1800     PAIR_CLASS_INIT(&tag, "cmd");
1801     print_otag(h, TAG_B, 1, &tag);
1802     return(1);
1803 }

1806 /* ARGSUSED */
1807 static int
1808 mdoc_rv_pre(MDOC_ARGS)
1809 {
1810     struct htmlpair tag;
1811     struct tag      *t;
1812     int             nchild;

1814     if (n->prev)
1815         print_otag(h, TAG_BR, 0, NULL);

1817     PAIR_CLASS_INIT(&tag, "fname");

1819     print_text(h, "The");

1821     nchild = n->nchild;
1822     for (n = n->child; n; n = n->next) {
1823         assert(MDOC_TEXT == n->type);

1825         t = print_otag(h, TAG_B, 1, &tag);
1826         print_text(h, n->string);
1827         print_tagq(h, t);

1829         h->flags |= HTML_NOSPACE;
1830         print_text(h, "()");

1832         if (nchild > 2 && n->next) {
1833             h->flags |= HTML_NOSPACE;
1834             print_text(h, ",");
1835         }

1837         if (n->next && NULL == n->next->next)
1838             print_text(h, "and");

```

```

1839     }
1841     if (nchild > 1)
1842         print_text(h, "functions return");
1843     else
1844         print_text(h, "function returns");
1846     print_text(h, "the value 0 if successful; otherwise the value "
1847               "-1 is returned and the global variable");
1849     PAIR_CLASS_INIT(&tag, "var");
1850     t = print_otag(h, TAG_B, 1, &tag);
1851     print_text(h, "errno");
1852     print_tagq(h, t);
1853     print_text(h, "is set to indicate the error.");
1854     return(0);
1855 }

1858 /* ARGSUSED */
1859 static int
1860 mdoc_va_pre(MDOC_ARGS)
1861 {
1862     struct htmlpair tag;
1864     PAIR_CLASS_INIT(&tag, "var");
1865     print_otag(h, TAG_B, 1, &tag);
1866     return(1);
1867 }

1870 /* ARGSUSED */
1871 static int
1872 mdoc_ap_pre(MDOC_ARGS)
1873 {
1874
1875     h->flags |= HTML_NOSPACE;
1876     print_text(h, "\\(aq");
1877     h->flags |= HTML_NOSPACE;
1878     return(1);
1879 }

1882 /* ARGSUSED */
1883 static int
1884 mdoc_bf_pre(MDOC_ARGS)
1885 {
1886     struct htmlpair tag[2];
1887     struct roffsu su;
1889     if (MDOC_HEAD == n->type)
1890         return(0);
1891     else if (MDOC_BODY != n->type)
1892         return(1);
1894     if (FONT_Em == n->norm->Bf.font)
1895         PAIR_CLASS_INIT(&tag[0], "emph");
1896     else if (FONT_Sy == n->norm->Bf.font)
1897         PAIR_CLASS_INIT(&tag[0], "symb");
1898     else if (FONT_Li == n->norm->Bf.font)
1899         PAIR_CLASS_INIT(&tag[0], "lit");
1900     else
1901         PAIR_CLASS_INIT(&tag[0], "none");
1903     /*
1904     * We want this to be inline-formatted, but needs to be div to

```

```

1905     * accept block children.
1906     */
1907     bufinit(h);
1908     bufcat_style(h, "display", "inline");
1909     SCALE_HS_INIT(&su, 1);
1910     /* Needs a left-margin for spacing. */
1911     bufcat_su(h, "margin-left", &su);
1912     PAIR_STYLE_INIT(&tag[1], h);
1913     print_otag(h, TAG_DIV, 2, tag);
1914     return(1);
1915 }

1918 /* ARGSUSED */
1919 static int
1920 mdoc_ms_pre(MDOC_ARGS)
1921 {
1922     struct htmlpair tag;
1924     PAIR_CLASS_INIT(&tag, "symb");
1925     print_otag(h, TAG_SPAN, 1, &tag);
1926     return(1);
1927 }

1930 /* ARGSUSED */
1931 static int
1932 mdoc_igndelim_pre(MDOC_ARGS)
1933 {
1935     h->flags |= HTML_IGNDELIM;
1936     return(1);
1937 }

1940 /* ARGSUSED */
1941 static void
1942 mdoc_pf_post(MDOC_ARGS)
1943 {
1945     h->flags |= HTML_NOSPACE;
1946 }

1949 /* ARGSUSED */
1950 static int
1951 mdoc_rs_pre(MDOC_ARGS)
1952 {
1953     struct htmlpair tag;
1955     if (MDOC_BLOCK != n->type)
1956         return(1);
1958     if (n->prev && SEC_SEE_ALSO == n->sec)
1959         print_otag(h, TAG_P, 0, NULL);
1961     PAIR_CLASS_INIT(&tag, "ref");
1962     print_otag(h, TAG_SPAN, 1, &tag);
1963     return(1);
1964 }

1968 /* ARGSUSED */
1969 static int
1970 mdoc_li_pre(MDOC_ARGS)

```

```

1971 {
1972     struct htmlpair tag;

1974     PAIR_CLASS_INIT(&tag, "lit");
1975     print_otag(h, TAG_CODE, 1, &tag);
1976     return(1);
1977 }

1980 /* ARGSUSED */
1981 static int
1982 mdoc_sy_pre(MDOC_ARGS)
1983 {
1984     struct htmlpair tag;

1986     PAIR_CLASS_INIT(&tag, "symb");
1987     print_otag(h, TAG_SPAN, 1, &tag);
1988     return(1);
1989 }

1992 /* ARGSUSED */
1993 static int
1994 mdoc_bt_pre(MDOC_ARGS)
1995 {

1997     print_text(h, "is currently in beta test.");
1998     return(0);
1999 }

2002 /* ARGSUSED */
2003 static int
2004 mdoc_ud_pre(MDOC_ARGS)
2005 {

2007     print_text(h, "currently under development.");
2008     return(0);
2009 }

2012 /* ARGSUSED */
2013 static int
2014 mdoc_lb_pre(MDOC_ARGS)
2015 {
2016     struct htmlpair tag;

2018     if (SEC_LIBRARY == n->sec && MDOC_LINE & n->flags && n->prev)
2019         print_otag(h, TAG_BR, 0, NULL);

2021     PAIR_CLASS_INIT(&tag, "lib");
2022     print_otag(h, TAG_SPAN, 1, &tag);
2023     return(1);
2024 }

2027 /* ARGSUSED */
2028 static int
2029 mdoc_x_pre(MDOC_ARGS)
2030 {
2031     struct htmlpair tag[2];
2032     enum htmltag t;

2034     t = TAG_SPAN;

2036     switch (n->tok) {

```

```

2037     case(MDOC__A):
2038         PAIR_CLASS_INIT(&tag[0], "ref-auth");
2039         if (n->prev && MDOC__A == n->prev->tok)
2040             if (NULL == n->next || MDOC__A != n->next->tok)
2041                 print_text(h, "and");
2042         break;
2043     case(MDOC__B):
2044         PAIR_CLASS_INIT(&tag[0], "ref-book");
2045         t = TAG_I;
2046         break;
2047     case(MDOC__C):
2048         PAIR_CLASS_INIT(&tag[0], "ref-city");
2049         break;
2050     case(MDOC__D):
2051         PAIR_CLASS_INIT(&tag[0], "ref-date");
2052         break;
2053     case(MDOC__I):
2054         PAIR_CLASS_INIT(&tag[0], "ref-issue");
2055         t = TAG_I;
2056         break;
2057     case(MDOC__J):
2058         PAIR_CLASS_INIT(&tag[0], "ref-jrnl");
2059         t = TAG_I;
2060         break;
2061     case(MDOC__N):
2062         PAIR_CLASS_INIT(&tag[0], "ref-num");
2063         break;
2064     case(MDOC__O):
2065         PAIR_CLASS_INIT(&tag[0], "ref-opt");
2066         break;
2067     case(MDOC__P):
2068         PAIR_CLASS_INIT(&tag[0], "ref-page");
2069         break;
2070     case(MDOC__Q):
2071         PAIR_CLASS_INIT(&tag[0], "ref-corp");
2072         break;
2073     case(MDOC__R):
2074         PAIR_CLASS_INIT(&tag[0], "ref-rep");
2075         break;
2076     case(MDOC__T):
2077         PAIR_CLASS_INIT(&tag[0], "ref-title");
2078         break;
2079     case(MDOC__U):
2080         PAIR_CLASS_INIT(&tag[0], "link-ref");
2081         break;
2082     case(MDOC__V):
2083         PAIR_CLASS_INIT(&tag[0], "ref-vol");
2084         break;
2085     default:
2086         abort();
2087         /* NOTREACHED */
2088     }

2090     if (MDOC__U != n->tok) {
2091         print_otag(h, t, 1, tag);
2092         return(1);
2093     }

2095     PAIR_HREF_INIT(&tag[1], n->child->string);
2096     print_otag(h, TAG_A, 2, tag);

2098     return(1);
2099 }

2102 /* ARGSUSED */

```

```

2103 static void
2104 mdoc__x_post(MDOC_ARGS)
2105 {
2107     if (MDOC_A == n->tok && n->next && MDOC_A == n->next->tok)
2108         if (NULL == n->next->next || MDOC_A != n->next->next->tok)
2109             if (NULL == n->prev || MDOC_A != n->prev->tok)
2110                 return;
2112     /* TODO: %U */
2114     if (NULL == n->parent || MDOC_Rs != n->parent->tok)
2115         return;
2117     h->flags |= HTML_NOSPACE;
2118     print_text(h, n->next ? ", " : ".");
2119 }
2122 /* ARGSUSED */
2123 static int
2124 mdoc_bk_pre(MDOC_ARGS)
2125 {
2127     switch (n->type) {
2128     case (MDOC_BLOCK):
2129         break;
2130     case (MDOC_HEAD):
2131         return(0);
2132     case (MDOC_BODY):
2133         if (n->parent->args || 0 == n->prev->nchild)
2134             h->flags |= HTML_PREKEEP;
2135         break;
2136     default:
2137         abort();
2138         /* NOTREACHED */
2139     }
2141     return(1);
2142 }
2145 /* ARGSUSED */
2146 static void
2147 mdoc_bk_post(MDOC_ARGS)
2148 {
2150     if (MDOC_BODY == n->type)
2151         h->flags &= ~(HTML_KEEP | HTML_PREKEEP);
2152 }
2155 /* ARGSUSED */
2156 static int
2157 mdoc_quote_pre(MDOC_ARGS)
2158 {
2159     struct htmlpair tag;
2161     if (MDOC_BODY != n->type)
2162         return(1);
2164     switch (n->tok) {
2165     case (MDOC_Ao):
2166         /* FALLTHROUGH */
2167     case (MDOC_Aq):
2168         print_text(h, "\\(la");

```

```

2169         break;
2170     case (MDOC_Bro):
2171         /* FALLTHROUGH */
2172     case (MDOC_Brq):
2173         print_text(h, "\\(lc");
2174         break;
2175     case (MDOC_Bo):
2176         /* FALLTHROUGH */
2177     case (MDOC_Bq):
2178         print_text(h, "\\(lb");
2179         break;
2180     case (MDOC_Oo):
2181         /* FALLTHROUGH */
2182     case (MDOC_Op):
2183         print_text(h, "\\(lb");
2184         h->flags |= HTML_NOSPACE;
2185         PAIR_CLASS_INIT(&tag, "opt");
2186         print_otag(h, TAG_SPAN, 1, &tag);
2187         break;
2188     case (MDOC_Eo):
2189         break;
2190     case (MDOC_Do):
2191         /* FALLTHROUGH */
2192     case (MDOC_Dq):
2193         /* FALLTHROUGH */
2194     case (MDOC_Qo):
2195         /* FALLTHROUGH */
2196     case (MDOC_Qq):
2197         print_text(h, "\\(lq");
2198         break;
2199     case (MDOC_Po):
2200         /* FALLTHROUGH */
2201     case (MDOC_Pq):
2202         print_text(h, "(");
2203         break;
2204     case (MDOC_Ql):
2205         print_text(h, "\\(oq");
2206         h->flags |= HTML_NOSPACE;
2207         PAIR_CLASS_INIT(&tag, "lit");
2208         print_otag(h, TAG_CODE, 1, &tag);
2209         break;
2210     case (MDOC_So):
2211         /* FALLTHROUGH */
2212     case (MDOC_Sq):
2213         print_text(h, "\\(oq");
2214         break;
2215     default:
2216         abort();
2217         /* NOTREACHED */
2218     }
2220     h->flags |= HTML_NOSPACE;
2221     return(1);
2222 }
2225 /* ARGSUSED */
2226 static void
2227 mdoc_quote_post(MDOC_ARGS)
2228 {
2230     if (MDOC_BODY != n->type)
2231         return;
2233     h->flags |= HTML_NOSPACE;

```



```
2235     switch (n->tok) {
2236     case (MDOC_Ao):
2237         /* FALLTHROUGH */
2238     case (MDOC_Aq):
2239         print_text(h, "\\(ra");
2240         break;
2241     case (MDOC_Bro):
2242         /* FALLTHROUGH */
2243     case (MDOC_Brq):
2244         print_text(h, "\\(rC");
2245         break;
2246     case (MDOC_Oo):
2247         /* FALLTHROUGH */
2248     case (MDOC_Op):
2249         /* FALLTHROUGH */
2250     case (MDOC_Bo):
2251         /* FALLTHROUGH */
2252     case (MDOC_Bq):
2253         print_text(h, "\\(rB");
2254         break;
2255     case (MDOC_Eo):
2256         break;
2257     case (MDOC_Qo):
2258         /* FALLTHROUGH */
2259     case (MDOC_Qq):
2260         /* FALLTHROUGH */
2261     case (MDOC_Do):
2262         /* FALLTHROUGH */
2263     case (MDOC_Dq):
2264         print_text(h, "\\(rq");
2265         break;
2266     case (MDOC_Po):
2267         /* FALLTHROUGH */
2268     case (MDOC_Pq):
2269         print_text(h, ")");
2270         break;
2271     case (MDOC_Ql):
2272         /* FALLTHROUGH */
2273     case (MDOC_So):
2274         /* FALLTHROUGH */
2275     case (MDOC_Sq):
2276         print_text(h, "\\(aq");
2277         break;
2278     default:
2279         abort();
2280         /* NOTREACHED */
2281     }
2282 }
```

```

*****
42928 Sat Jul 19 19:24:31 2014
new/usr/src/cmd/mandoc/mdoc_macro.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: mdoc_macro.c,v 1.115 2012/01/05 00:43:51 schwarze Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2010 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifndef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <assert.h>
23 #include <ctype.h>
24 #include <stdlib.h>
25 #include <stdio.h>
26 #include <string.h>
27 #include <time.h>
28
29 #include "mdoc.h"
30 #include "mandoc.h"
31 #include "libmdoc.h"
32 #include "libmandoc.h"
33
34 enum      rew { /* see rew_dohalt() */
35     REWIND_NONE,
36     REWIND_THIS,
37     REWIND_MORE,
38     REWIND_FORCE,
39     REWIND_LATER,
40     REWIND_ERROR
41 };
42
43 static int      blk_full(MACRO_PROT_ARGS);
44 static int      blk_exp_close(MACRO_PROT_ARGS);
45 static int      blk_part_exp(MACRO_PROT_ARGS);
46 static int      blk_part_imp(MACRO_PROT_ARGS);
47 static int      ctx_synopsis(MACRO_PROT_ARGS);
48 static int      in_line_eoln(MACRO_PROT_ARGS);
49 static int      in_line_argn(MACRO_PROT_ARGS);
50 static int      in_line(MACRO_PROT_ARGS);
51 static int      obsolete(MACRO_PROT_ARGS);
52 static int      phrase_ta(MACRO_PROT_ARGS);
53
54 static int      dword(struct mdoc *, int, int,
55                     const char *, enum mdelim);
56 static int      append_delims(struct mdoc *,

```

```

int, int *, char *);
58 static enum mdoc      lookup(enum mdoc, const char *);
59 static enum mdoc      lookup_raw(const char *);
60 static int             make_pending(struct mdoc_node *, enum mdoc,
61                                 struct mdoc *, int, int);
62 static int             phrase(struct mdoc *, int, int, char *);
63 static enum mdoc      rew_alt(enum mdoc);
64 static enum rew       rew_dohalt(enum mdoc, enum mdoc_type,
65                                 const struct mdoc_node *);
66 static int            rew_elem(struct mdoc *, enum mdoc);
67 static int            rew_last(struct mdoc *);
68                                 const struct mdoc_node *);
69 static int            rew_sub(enum mdoc_type, struct mdoc *,
70                                 enum mdoc, int, int);
71
72 const      struct mdoc_macro __mdoc_macros[MDOC_MAX] = {
73     { in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* Ap */
74     { in_line_eoln, MDOC_PROLOGUE }, /* Dd */
75     { in_line_eoln, MDOC_PROLOGUE }, /* Dt */
76     { in_line_eoln, MDOC_PROLOGUE }, /* Os */
77     { blk_full, MDOC_PARSED }, /* Sh */
78     { blk_full, MDOC_PARSED }, /* Ss */
79     { in_line_eoln, 0 }, /* Pp */
80     { blk_part_imp, MDOC_PARSED }, /* D1 */
81     { blk_part_imp, MDOC_PARSED }, /* D1 */
82     { blk_full, MDOC_EXPLICIT }, /* Bd */
83     { blk_exp_close, MDOC_EXPLICIT }, /* Ed */
84     { blk_full, MDOC_EXPLICIT }, /* B1 */
85     { blk_exp_close, MDOC_EXPLICIT }, /* E1 */
86     { blk_full, MDOC_PARSED }, /* It */
87     { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Ad */
88     { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* An */
89     { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Ar */
90     { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Cd */
91     { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Cm */
92     { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Dv */
93     { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Er */
94     { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Ev */
95     { in_line_eoln, 0 }, /* Ex */
96     { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Fa */
97     { in_line_eoln, 0 }, /* Fd */
98     { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Fl */
99     { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Fn */
100    { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Ft */
101    { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Ic */
102    { in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* In */
103    { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Li */
104    { blk_full, 0 }, /* Nd */
105    { ctx_synopsis, MDOC_CALLABLE | MDOC_PARSED }, /* Nm */
106    { blk_part_imp, MDOC_CALLABLE | MDOC_PARSED }, /* Op */
107    { obsolete, 0 }, /* Ot */
108    { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Pa */
109    { in_line_eoln, 0 }, /* Rv */
110    { in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* St */
111    { in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Va */
112    { ctx_synopsis, MDOC_CALLABLE | MDOC_PARSED }, /* Vt */
113    { in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* Xr */
114    { in_line_eoln, 0 }, /* %A */
115    { in_line_eoln, 0 }, /* %B */
116    { in_line_eoln, 0 }, /* %D */
117    { in_line_eoln, 0 }, /* %I */
118    { in_line_eoln, 0 }, /* %J */
119    { in_line_eoln, 0 }, /* %N */
120    { in_line_eoln, 0 }, /* %O */
121    { in_line_eoln, 0 }, /* %P */
122    { in_line_eoln, 0 }, /* %R */

```

```

123 { in_line_eoln, 0 }, /* %T */
124 { in_line_eoln, 0 }, /* %V */
125 blk_exp_close, MDOC_EXPLICIT | MDOC_CALLABLE | MDOC_PARSED }, /* Ac */
126 blk_part_exp, MDOC_CALLABLE | MDOC_PARSED | MDOC_EXPLICIT }, /* Ao */
127 blk_part_imp, MDOC_CALLABLE | MDOC_PARSED }, /* Aq */
128 in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* At */
129 blk_exp_close, MDOC_EXPLICIT | MDOC_CALLABLE | MDOC_PARSED }, /* Bc */
130 blk_full, MDOC_EXPLICIT }, /* Bf */
131 blk_part_exp, MDOC_CALLABLE | MDOC_PARSED | MDOC_EXPLICIT }, /* Bo */
132 blk_part_imp, MDOC_CALLABLE | MDOC_PARSED }, /* Bq */
133 in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* Bsx */
134 in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* Bx */
135 { in_line_eoln, 0 }, /* Db */
136 blk_exp_close, MDOC_EXPLICIT | MDOC_CALLABLE | MDOC_PARSED }, /* Dc */
137 blk_part_exp, MDOC_CALLABLE | MDOC_PARSED | MDOC_EXPLICIT }, /* Do */
138 blk_part_imp, MDOC_CALLABLE | MDOC_PARSED }, /* Dq */
139 blk_exp_close, MDOC_EXPLICIT | MDOC_CALLABLE | MDOC_PARSED }, /* Ec */
140 blk_exp_close, MDOC_EXPLICIT }, /* Ef */
141 in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Em */
142 blk_part_exp, MDOC_CALLABLE | MDOC_PARSED | MDOC_EXPLICIT }, /* Eo */
143 in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* Fx */
144 in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Ms */
145 in_line_argn, MDOC_CALLABLE | MDOC_PARSED | MDOC_IGNDELIM }, /* No */
146 in_line_argn, MDOC_CALLABLE | MDOC_PARSED | MDOC_IGNDELIM }, /* Ns */
147 in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* Nx */
148 in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* Ox */
149 blk_exp_close, MDOC_EXPLICIT | MDOC_CALLABLE | MDOC_PARSED }, /* Pc */
150 in_line_argn, MDOC_CALLABLE | MDOC_PARSED | MDOC_IGNDELIM }, /* Pf */
151 blk_part_exp, MDOC_CALLABLE | MDOC_PARSED | MDOC_EXPLICIT }, /* Po */
152 blk_part_imp, MDOC_CALLABLE | MDOC_PARSED }, /* Pq */
153 blk_exp_close, MDOC_EXPLICIT | MDOC_CALLABLE | MDOC_PARSED }, /* Qc */
154 blk_part_imp, MDOC_CALLABLE | MDOC_PARSED }, /* Ql */
155 blk_part_exp, MDOC_CALLABLE | MDOC_PARSED | MDOC_EXPLICIT }, /* Qo */
156 blk_part_imp, MDOC_CALLABLE | MDOC_PARSED }, /* Qq */
157 blk_exp_close, MDOC_EXPLICIT }, /* Re */
158 blk_full, MDOC_EXPLICIT }, /* Rs */
159 blk_exp_close, MDOC_EXPLICIT | MDOC_CALLABLE | MDOC_PARSED }, /* Sc */
160 blk_part_exp, MDOC_CALLABLE | MDOC_PARSED | MDOC_EXPLICIT }, /* So */
161 blk_part_imp, MDOC_CALLABLE | MDOC_PARSED }, /* Sq */
162 in_line_eoln, 0 }, /* Sm */
163 in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Sx */
164 in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Sy */
165 in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Tn */
166 in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* Ux */
167 blk_exp_close, MDOC_EXPLICIT | MDOC_CALLABLE | MDOC_PARSED }, /* Xc */
168 blk_part_exp, MDOC_CALLABLE | MDOC_PARSED | MDOC_EXPLICIT }, /* Xo */
169 blk_full, MDOC_EXPLICIT | MDOC_CALLABLE }, /* Fo */
170 blk_exp_close, MDOC_EXPLICIT | MDOC_CALLABLE | MDOC_PARSED }, /* Fc */
171 blk_part_exp, MDOC_CALLABLE | MDOC_PARSED | MDOC_EXPLICIT }, /* Oo */
172 blk_exp_close, MDOC_EXPLICIT | MDOC_CALLABLE | MDOC_PARSED }, /* Oc */
173 blk_full, MDOC_EXPLICIT }, /* Bk */
174 blk_exp_close, MDOC_EXPLICIT }, /* Ek */
175 { in_line_eoln, 0 }, /* Bt */
176 { in_line_eoln, 0 }, /* Hf */
177 obsolete, 0 }, /* Fr */
178 { in_line_eoln, 0 }, /* Ud */
179 { in_line, 0 }, /* Lb */
180 { in_line_eoln, 0 }, /* Lp */
181 in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Lk */
182 in_line, MDOC_CALLABLE | MDOC_PARSED }, /* Mt */
183 blk_part_imp, MDOC_CALLABLE | MDOC_PARSED }, /* Brq */
184 blk_part_exp, MDOC_CALLABLE | MDOC_PARSED | MDOC_EXPLICIT }, /* Bro */
185 blk_exp_close, MDOC_EXPLICIT | MDOC_CALLABLE | MDOC_PARSED }, /* Brc */
186 { in_line_eoln, 0 }, /* %C */
187 obsolete, 0 }, /* Es */
188 obsolete, 0 }, /* En */

```

```

189 { in_line_argn, MDOC_CALLABLE | MDOC_PARSED }, /* Dx */
190 { in_line_eoln, 0 }, /* %Q */
191 { in_line_eoln, 0 }, /* br */
192 { in_line_eoln, 0 }, /* sp */
193 { in_line_eoln, 0 }, /* %U */
194 { phrase_ta, MDOC_CALLABLE | MDOC_PARSED }, /* Ta */
195 };

197 const struct mdoc_macro * const mdoc_macros = __mdoc_macros;

200 /*
201 * This is called at the end of parsing. It must traverse up the tree,
202 * closing out open [implicit] scopes. Obviously, open explicit scopes
203 * are errors.
204 */
205 int
206 mdoc_macroend(struct mdoc *m)
207 {
208     struct mdoc_node *n;

210     /* Scan for open explicit scopes. */

212     n = MDOC_VALID & m->last->flags ? m->last->parent : m->last;

214     for ( ; n; n = n->parent)
215         if (MDOC_BLOCK == n->type &&
216             MDOC_EXPLICIT & mdoc_macros[n->tok].flags)
217             mdoc_nmsg(m, n, MANDOCERR_SCOPEEXIT);

219     /* Rewind to the first. */

221     return(rew_last(m, m->first));
222 }

225 /*
226 * Look up a macro from within a subsequent context.
227 */
228 static enum mdoct
229 lookup(enum mdoct from, const char *p)
230 {

232     if ( ! (MDOC_PARSED & mdoc_macros[from].flags) )
233         return(MDOC_MAX);
234     return(lookup_raw(p));
235 }

238 /*
239 * Lookup a macro following the initial line macro.
240 */
241 static enum mdoct
242 lookup_raw(const char *p)
243 {
244     enum mdoct     res;

246     if (MDOC_MAX == (res = mdoc_hash_find(p)))
247         return(MDOC_MAX);
248     if (MDOC_CALLABLE & mdoc_macros[res].flags)
249         return(res);
250     return(MDOC_MAX);
251 }

254 static int

```

```

255 rew_last(struct mdoc *mdoc, const struct mdoc_node *to)
256 {
257     struct mdoc_node *n, *np;

259     assert(to);
260     mdoc->next = MDOC_NEXT_SIBLING;

262     /* LINTED */
263     while (mdoc->last != to) {
264         /*
265          * Save the parent here, because we may delete the
266          * m->last node in the post-validation phase and reset
267          * it to m->last->parent, causing a step in the closing
268          * out to be lost.
269          */
270         np = mdoc->last->parent;
271         if ( ! mdoc_valid_post(mdoc))
272             return(0);
273         n = mdoc->last;
274         mdoc->last = np;
275         assert(mdoc->last);
276         mdoc->last->last = n;
277     }

279     return(mdoc_valid_post(mdoc));
280 }

283 /*
284  * For a block closing macro, return the corresponding opening one.
285  * Otherwise, return the macro itself.
286  */
287 static enum mdoct
288 rew_alt(enum mdoct tok)
289 {
290     switch (tok) {
291     case (MDOC_Ac):
292         return(MDOC_Ao);
293     case (MDOC_Bc):
294         return(MDOC_Bo);
295     case (MDOC_Brc):
296         return(MDOC_Bro);
297     case (MDOC_Dc):
298         return(MDOC_Do);
299     case (MDOC_Ec):
300         return(MDOC_Eo);
301     case (MDOC_Ed):
302         return(MDOC_Bd);
303     case (MDOC_Ef):
304         return(MDOC_Bf);
305     case (MDOC_Ek):
306         return(MDOC_Bk);
307     case (MDOC_El):
308         return(MDOC_Bl);
309     case (MDOC_Fc):
310         return(MDOC_Fo);
311     case (MDOC_Oc):
312         return(MDOC_Oo);
313     case (MDOC_Pc):
314         return(MDOC_Po);
315     case (MDOC_Qc):
316         return(MDOC_Qo);
317     case (MDOC_Re):
318         return(MDOC_Rs);
319     case (MDOC_Sc):
320         return(MDOC_So);

```

```

321     case (MDOC_Xc):
322         return(MDOC_Xo);
323     default:
324         return(tok);
325     }
326     /* NOTREACHED */
327 }

330 /*
331  * Rewinding to tok, how do we have to handle *p?
332  * REWIND_NONE: *p would delimit tok, but no tok scope is open
333  *   inside *p, so there is no need to rewind anything at all.
334  * REWIND_THIS: *p matches tok, so rewind *p and nothing else.
335  * REWIND_MORE: *p is implicit, rewind it and keep searching for tok.
336  * REWIND_FORCE: *p is explicit, but tok is full, force rewinding *p.
337  * REWIND_LATER: *p is explicit and still open, postpone rewinding.
338  * REWIND_ERROR: No tok block is open at all.
339  */
340 static enum rew
341 rew_dohalt(enum mdoct tok, enum mdoc_type type,
342           const struct mdoc_node *p)
343 {
344
345     /*
346      * No matching token, no delimiting block, no broken block.
347      * This can happen when full implicit macros are called for
348      * the first time but try to rewind their previous
349      * instance anyway.
350      */
351     if (MDOC_ROOT == p->type)
352         return(MDOC_BLOCK == type &&
353                MDOC_EXPLICIT & mdoc_macros[tok].flags ?
354                REWIND_ERROR : REWIND_NONE);
355
356     /*
357      * When starting to rewind, skip plain text
358      * and nodes that have already been rewound.
359      */
360     if (MDOC_TEXT == p->type || MDOC_VALID & p->flags)
361         return(REWIND_MORE);
362
363     /*
364      * The easiest case: Found a matching token.
365      * This applies to both blocks and elements.
366      */
367     tok = rew_alt(tok);
368     if (tok == p->tok)
369         return(p->end ? REWIND_NONE :
370                type == p->type ? REWIND_THIS : REWIND_MORE);
371
372     /*
373      * While elements do require rewinding for themselves,
374      * they never affect rewinding of other nodes.
375      */
376     if (MDOC_ELEM == p->type)
377         return(REWIND_MORE);
378
379     /*
380      * Blocks delimited by our target token get REWIND_MORE.
381      * Blocks delimiting our target token get REWIND_NONE.
382      */
383     switch (tok) {
384     case (MDOC_Bl):
385         if (MDOC_It == p->tok)
386             return(REWIND_MORE);

```

```

387     break;
388 case (MDOC_It):
389     if (MDOC_BODY == p->type && MDOC_Bl == p->tok)
390         return(REWIND_NONE);
391     break;
392 /*
393  * XXX Badly nested block handling still fails badly
394  * when one block is breaking two blocks of the same type.
395  * This is an incomplete and extremely ugly workaround,
396  * required to let the OpenBSD tree build.
397  */
398 case (MDOC_Oo):
399     if (MDOC_Op == p->tok)
400         return(REWIND_MORE);
401     break;
402 case (MDOC_Nm):
403     return(REWIND_NONE);
404 case (MDOC_Nd):
405     /* FALLTHROUGH */
406 case (MDOC_Ss):
407     if (MDOC_BODY == p->type && MDOC_Sh == p->tok)
408         return(REWIND_NONE);
409     /* FALLTHROUGH */
410 case (MDOC_Sh):
411     if (MDOC_Nd == p->tok || MDOC_Ss == p->tok ||
412         MDOC_Sh == p->tok)
413         return(REWIND_MORE);
414     break;
415 default:
416     break;
417 }
418
419 /*
420  * Default block rewinding rules.
421  * In particular, always skip block end markers,
422  * and let all blocks rewind Nm children.
423  */
424 if (ENDBODY_NOT != p->end || MDOC_Nm == p->tok ||
425     (MDOC_BLOCK == p->type &&
426      ! (MDOC_EXPLICIT & mdoc_macros[tok].flags)))
427     return(REWIND_MORE);
428
429 /*
430  * By default, closing out full blocks
431  * forces closing of broken explicit blocks,
432  * while closing out partial blocks
433  * allows delayed rewinding by default.
434  */
435 return (&blk_full == mdoc_macros[tok].fp ?
436     REWIND_FORCE : REWIND_LATER);
437 }
438
439
440 static int
441 rew_elem(struct mdoc *mdoc, enum mdoct tok)
442 {
443     struct mdoc_node *n;
444
445     n = mdoc->last;
446     if (MDOC_ELEM != n->type)
447         n = n->parent;
448     assert(MDOC_ELEM == n->type);
449     assert(tok == n->tok);
450
451     return(rew_last(mdoc, n));
452 }

```

```

455 /*
456  * We are trying to close a block identified by tok,
457  * but the child block *broken is still open.
458  * Thus, postpone closing the tok block
459  * until the rew_sub call closing *broken.
460  */
461 static int
462 make_pending(struct mdoc_node *broken, enum mdoct tok,
463             struct mdoc *m, int line, int ppos)
464 {
465     struct mdoc_node *breaker;
466
467     /*
468      * Iterate backwards, searching for the block matching tok,
469      * that is, the block breaking the *broken block.
470      */
471     for (breaker = broken->parent; breaker; breaker = breaker->parent) {
472
473         /*
474          * If the *broken block had already been broken before
475          * and we encounter its breaker, make the tok block
476          * pending on the inner breaker.
477          * Graphically, "[A breaker=[B broken=[C->B B] tok=A] C]"
478          * becomes "[A broken=[B [C->B B] tok=A] C]"
479          * and finally "[A [B->A [C->B B] A] C]".
480          */
481         if (breaker == broken->pending) {
482             broken = breaker;
483             continue;
484         }
485
486         if (REWIND_THIS != rew_dohalt(tok, MDOC_BLOCK, breaker))
487             continue;
488         if (MDOC_BODY == broken->type)
489             broken = broken->parent;
490
491         /*
492          * Found the breaker.
493          * If another, outer breaker is already pending on
494          * the *broken block, we must not clobber the link
495          * to the outer breaker, but make it pending on the
496          * new, now inner breaker.
497          * Graphically, "[A breaker=[B broken=[C->A A] tok=B] C]"
498          * becomes "[A breaker=[B->A broken=[C A] tok=B] C]"
499          * and finally "[A [B->A [C->B A] B] C]".
500          */
501         if (broken->pending) {
502             struct mdoc_node *taker;
503
504             /*
505              * If the breaker had also been broken before,
506              * it cannot take on the outer breaker itself,
507              * but must hand it on to its own breakers.
508              * Graphically, this is the following situation:
509              * "[A [B breaker=[C->B B] broken=[D->A A] tok=C] D]"
510              * "[A taker=[B->A breaker=[C->B B] [D->C A] C] D]"
511              */
512             taker = breaker;
513             while (taker->pending)
514                 taker = taker->pending;
515             taker->pending = broken->pending;
516         }
517         broken->pending = breaker;
518         mandoc_vmsg(MANDOCERR_SCOPENEST, m->parse, line, ppos,

```

```

519         "%s breaks %s", mdoc_macronames[tok],
520         mdoc_macronames[broken->tok]);
521     }
522     return(1);
523 }
524 /*
525  * Found no matching block for tok.
526  * Are you trying to close a block that is not open?
527  */
528 return(0);
529 }

532 static int
533 rew_sub(enum mdoc_type t, struct mdoc *m,
534         enum mdoct tok, int line, int ppos)
535 {
536     struct mdoc_node *n;

538     n = m->last;
539     while (n) {
540         switch (rew_dohalt(tok, t, n)) {
541             case (REWIND_NONE):
542                 return(1);
543             case (REWIND_THIS):
544                 break;
545             case (REWIND_FORCE):
546                 mandoc_vmsg(MANDOCERR_SCOPEBROKEN, m->parse,
547                             line, ppos, "%s breaks %s",
548                             mdoc_macronames[tok],
549                             mdoc_macronames[n->tok]);
550                 /* FALLTHROUGH */
551             case (REWIND_MORE):
552                 n = n->parent;
553                 continue;
554             case (REWIND_LATER):
555                 if (make_pending(n, tok, m, line, ppos) ||
556                     MDOC_BLOCK != t)
557                     return(1);
558                 /* FALLTHROUGH */
559             case (REWIND_ERROR):
560                 mdoc_pmsg(m, line, ppos, MANDOCERR_NOSCOPE);
561                 return(1);
562         }
563         break;
564     }

566     assert(n);
567     if (!rew_last(m, n))
568         return(0);

570     /*
571      * The current block extends an enclosing block.
572      * Now that the current block ends, close the enclosing block, too.
573      */
574     while (NULL != (n = n->pending)) {
575         if (!rew_last(m, n))
576             return(0);
577         if (MDOC_HEAD == n->type &&
578             !mdoc_body_alloc(m, n->line, n->pos, n->tok))
579             return(0);
580     }

582     return(1);
583 }

```

```

585 /*
586  * Allocate a word and check whether it's punctuation or not.
587  * Punctuation consists of those tokens found in mdoc_isdelim().
588  */
589 static int
590 dword(struct mdoc *m, int line,
591        int col, const char *p, enum mdelim d)
592 {
593     if (DELIM_MAX == d)
594         d = mdoc_isdelim(p);

597     if (!mdoc_word_alloc(m, line, col, p))
598         return(0);

600     if (DELIM_OPEN == d)
601         m->last->flags |= MDOC_DELIMO;

603     /*
604      * Closing delimiters only suppress the preceding space
605      * when they follow something, not when they start a new
606      * block or element, and not when they follow 'No'.
607      *
608      * XXX Explicitly special-casing MDOC_No here feels
609      * like a layering violation. Find a better way
610      * and solve this in the code related to 'No'!
611      */

613     else if (DELIM_CLOSE == d && m->last->prev &&
614             m->last->prev->tok != MDOC_No)
615         m->last->flags |= MDOC_DELIMC;

617     return(1);
618 }

620 static int
621 append_delims(struct mdoc *m, int line, int *pos, char *buf)
622 {
623     int la;
624     enum margserr ac;
625     char *p;

627     if ('\0' == buf[*pos])
628         return(1);

630     for (;;) {
631         la = *pos;
632         ac = mdoc_zargs(m, line, pos, buf, &p);

634         if (ARGS_ERROR == ac)
635             return(0);
636         else if (ARGS_EOLN == ac)
637             break;

639         dword(m, line, la, p, DELIM_MAX);

641     /*
642      * If we encounter end-of-sentence symbols, then trigger
643      * the double-space.
644      *
645      * XXX: it's easy to allow this to propagate outward to
646      * the last symbol, such that '.' will cause the
647      * correct double-spacing. However, (1) groff isn't
648      * smart enough to do this and (2) it would require
649      * knowing which symbols break this behaviour, for
650      * example, '.' shouldn't propagate the double-space.

```

```

651         */
652         if (mandoc_eos(p, strlen(p), 0))
653             m->last->flags |= MDOC_EOS;
654     }
655
656     return(1);
657 }
658
659 /*
660  * Close out block partial/full explicit.
661  */
662 static int
663 blk_exp_close(MACRO_PROT_ARGS)
664 {
665     struct mdoc_node *body;          /* Our own body. */
666     struct mdoc_node *later;        /* A sub-block starting later. */
667     struct mdoc_node *n;           /* For searching backwards. */
668
669     int j, lastarg, maxargs, flushed, nl;
670     enum margserr ac;
671     enum mdoct atok, ntok;
672     char *p;
673
674     nl = MDOC_NEWLINE & m->flags;
675
676     switch (tok) {
677     case (MDOC_Ec):
678         maxargs = 1;
679         break;
680     default:
681         maxargs = 0;
682         break;
683     }
684
685     /*
686      * Search backwards for beginnings of blocks,
687      * both of our own and of pending sub-blocks.
688      */
689     atok = rew_alt(tok);
690     body = later = NULL;
691     for (n = m->last; n; n = n->parent) {
692         if (MDOC_VALID & n->flags)
693             continue;
694
695         /* Remember the start of our own body. */
696         if (MDOC_BODY == n->type && atok == n->tok) {
697             if (ENDBODY_NOT == n->end)
698                 body = n;
699             continue;
700         }
701     }
702
703     if (MDOC_BLOCK != n->type || MDOC_Nm == n->tok)
704         continue;
705     if (atok == n->tok) {
706         assert(body);
707
708         /*
709          * Found the start of our own block.
710          * When there is no pending sub block,
711          * just proceed to closing out.
712          */
713         if (NULL == later)
714             break;
715
716         /*

```

```

717         * When there is a pending sub block,
718         * postpone closing out the current block
719         * until the rew_sub() closing out the sub-block.
720         */
721         make_pending(later, tok, m, line, ppos);
722
723         /*
724          * Mark the place where the formatting - but not
725          * the scope - of the current block ends.
726          */
727         if (!mdoc_endbody_alloc(m, line, ppos,
728             atok, body, ENDBODY_SPACE))
729             return(0);
730         break;
731     }
732
733     /*
734      * When finding an open sub block, remember the last
735      * open explicit block, or, in case there are only
736      * implicit ones, the first open implicit block.
737      */
738     if (later &&
739         MDOC_EXPLICIT & mdoc_macros[later->tok].flags)
740         continue;
741     if (MDOC_CALLABLE & mdoc_macros[n->tok].flags)
742         later = n;
743
744     if (! (MDOC_CALLABLE & mdoc_macros[tok].flags)) {
745         /* FIXME: do this in validate */
746         if (buf[*pos])
747             mdoc_pmsg(m, line, ppos, MANDOCERR_ARGSLOST);
748
749         if (!rew_sub(MDOC_BODY, m, tok, line, ppos))
750             return(0);
751         return(rew_sub(MDOC_BLOCK, m, tok, line, ppos));
752     }
753
754     if (!rew_sub(MDOC_BODY, m, tok, line, ppos))
755         return(0);
756
757     if (NULL == later && maxargs > 0)
758         if (!mdoc_tail_alloc(m, line, ppos, rew_alt(tok)))
759             return(0);
760
761     for (flushed = j = 0; ; j++) {
762         lastarg = *pos;
763
764         if (j == maxargs && !flushed) {
765             if (!rew_sub(MDOC_BLOCK, m, tok, line, ppos))
766                 return(0);
767             flushed = 1;
768         }
769     }
770
771     ac = mdoc_args(m, line, pos, buf, tok, &p);
772
773     if (ARGS_ERROR == ac)
774         return(0);
775     if (ARGS_PUNCT == ac)
776         break;
777     if (ARGS_EOLN == ac)
778         break;
779
780     ntok = ARGS_QWORD == ac ? MDOC_MAX : lookup(tok, p);
781
782     if (MDOC_MAX == ntok) {

```

```

783         if ( ! dword(m, line, lastarg, p, DELIM_MAX))
784             return(0);
785         continue;
786     }
787
788     if ( ! flushed) {
789         if ( ! rew_sub(MDOC_BLOCK, m, tok, line, ppos))
790             return(0);
791         flushed = 1;
792     }
793     if ( ! mdoc_macro(m, ntok, line, lastarg, pos, buf))
794         return(0);
795     break;
796 }
797
798 if ( ! flushed && ! rew_sub(MDOC_BLOCK, m, tok, line, ppos))
799     return(0);
800
801 if ( ! nl)
802     return(1);
803 return(append_delims(m, line, pos, buf));
804 }
805
807 static int
808 in_line(MACRO_PROT_ARGS)
809 {
810     int             la, scope, cnt, nc, nl;
811     enum margverr   av;
812     enum mdoct      ntok;
813     enum margserr   ac;
814     enum mdelim     d;
815     struct mdoc_arg *arg;
816     char            *p;
817
818     nl = MDOC_NEWLINE & m->flags;
819
820     /*
821      * Whether we allow ignored elements (those without content,
822      * usually because of reserved words) to squeak by.
823      */
824
825     switch (tok) {
826     case (MDOC_An):
827         /* FALLTHROUGH */
828     case (MDOC_Ar):
829         /* FALLTHROUGH */
830     case (MDOC_Fl):
831         /* FALLTHROUGH */
832     case (MDOC_Mt):
833         /* FALLTHROUGH */
834     case (MDOC_Nm):
835         /* FALLTHROUGH */
836     case (MDOC_Pa):
837         nc = 1;
838         break;
839     default:
840         nc = 0;
841         break;
842     }
843
844     for (arg = NULL;; ) {
845         la = *pos;
846         av = mdoc_argv(m, line, tok, &arg, pos, buf);
847
848         if (ARGV_WORD == av) {

```

```

849             *pos = la;
850             break;
851         }
852         if (ARGV_EOLN == av)
853             break;
854         if (ARGV_ARG == av)
855             continue;
856
857         mdoc_argv_free(arg);
858         return(0);
859     }
860
861     for (cnt = scope = 0;; ) {
862         la = *pos;
863         ac = mdoc_args(m, line, pos, buf, tok, &p);
864
865         if (ARGS_ERROR == ac)
866             return(0);
867         if (ARGS_EOLN == ac)
868             break;
869         if (ARGS_PUNCT == ac)
870             break;
871
872         ntok = ARGS_QWORD == ac ? MDOC_MAX : lookup(tok, p);
873
874         /*
875          * In this case, we've located a submacro and must
876          * execute it. Close out scope, if open. If no
877          * elements have been generated, either create one (nc)
878          * or raise a warning.
879          */
880
881         if (MDOC_MAX != ntok) {
882             if (scope && ! rew_elem(m, tok))
883                 return(0);
884             if (nc && 0 == cnt) {
885                 if ( ! mdoc_elem_alloc(m, line, ppos, tok, arg))
886                     return(0);
887                 if ( ! rew_last(m, m->last))
888                     return(0);
889             } else if ( ! nc && 0 == cnt) {
890                 mdoc_argv_free(arg);
891                 mdoc_pmsg(m, line, ppos, MANDOCERR_MACROEMPTY);
892             }
893
894             if ( ! mdoc_macro(m, ntok, line, la, pos, buf))
895                 return(0);
896             if ( ! nl)
897                 return(1);
898             return(append_delims(m, line, pos, buf));
899         }
900
901         /*
902          * Non-quote-enclosed punctuation. Set up our scope, if
903          * a word; rewind the scope, if a delimiter; then append
904          * the word.
905          */
906
907         d = ARGS_QWORD == ac ? DELIM_NONE : mdoc_isdelim(p);
908
909         if (DELIM_NONE != d) {
910             /*
911              * If we encounter closing punctuation, no word
912              * has been omitted, no scope is open, and we're
913              * allowed to have an empty element, then start
914              * a new scope. 'Ar', 'Fl', and 'Li', only do

```



```

915     * this once per invocation.  There may be more
916     * of these (all of them?).
917     */
918     if (0 == cnt && (nc || MDOC_Li == tok) &&
919         DELIM_CLOSE == d && ! scope) {
920         if ( ! mdoc_elem_alloc(m, line, ppos, tok, arg))
921             return(0);
922         if (MDOC_Ar == tok || MDOC_Li == tok ||
923             MDOC_Fl == tok)
924             cnt++;
925         scope = 1;
926     }
927     /*
928     * Close out our scope, if one is open, before
929     * any punctuation.
930     */
931     if (scope && ! rew_elem(m, tok))
932         return(0);
933     scope = 0;
934 } else if ( ! scope) {
935     if ( ! mdoc_elem_alloc(m, line, ppos, tok, arg))
936         return(0);
937     scope = 1;
938 }
939
940 if (DELIM_NONE == d)
941     cnt++;
942
943 if ( ! dword(m, line, la, p, d))
944     return(0);
945
946 /*
947 * 'Fl' macros have their scope re-opened with each new
948 * word so that the '-' can be added to each one without
949 * having to parse out spaces.
950 */
951 if (scope && MDOC_Fl == tok) {
952     if ( ! rew_elem(m, tok))
953         return(0);
954     scope = 0;
955 }
956
957 if (scope && ! rew_elem(m, tok))
958     return(0);
959
960 /*
961 * If no elements have been collected and we're allowed to have
962 * empties (nc), open a scope and close it out.  Otherwise,
963 * raise a warning.
964 */
965
966 if (nc && 0 == cnt) {
967     if ( ! mdoc_elem_alloc(m, line, ppos, tok, arg))
968         return(0);
969     if ( ! rew_last(m, m->last))
970         return(0);
971 } else if ( ! nc && 0 == cnt) {
972     mdoc_argv_free(arg);
973     mdoc_pmsg(m, line, ppos, MANDOCERR_MACROEMPTY);
974 }
975
976 if ( ! nl)
977     return(1);
978 return(append_delims(m, line, pos, buf));
979 }

```

```

983 static int
984 blk_full(MACRO_PROT_ARGS)
985 {
986     int            la, nl, nparsed;
987     struct mdoc_arg *arg;
988     struct mdoc_node *head; /* save of head macro */
989     struct mdoc_node *body; /* save of body macro */
990     struct mdoc_node *n;
991     enum mdoc_type  mtt;
992     enum mdoct      ntok;
993     enum margser    ac, lac;
994     enum margverr   av;
995     char           *p;
996
997     nl = MDOC_NEWLINE & m->flags;
998
999     /* Close out prior implicit scope. */
1000
1001     if ( ! (MDOC_EXPLICIT & mdoc_macros[tok].flags)) {
1002         if ( ! rew_sub(MDOC_BODY, m, tok, line, ppos))
1003             return(0);
1004         if ( ! rew_sub(MDOC_BLOCK, m, tok, line, ppos))
1005             return(0);
1006     }
1007
1008     /*
1009     * This routine accommodates implicitly- and explicitly-scoped
1010     * macro openings.  Implicit ones first close out prior scope
1011     * (seen above).  Delay opening the head until necessary to
1012     * allow leading punctuation to print.  Special consideration
1013     * for 'It -column', which has phrase-part syntax instead of
1014     * regular child nodes.
1015     */
1016
1017     for (arg = NULL; ) {
1018         la = *pos;
1019         av = mdoc_argv(m, line, tok, &arg, pos, buf);
1020
1021         if (ARGV_WORD == av) {
1022             *pos = la;
1023             break;
1024         }
1025
1026         if (ARGV_EOLN == av)
1027             break;
1028         if (ARGV_ARG == av)
1029             continue;
1030
1031         mdoc_argv_free(arg);
1032         return(0);
1033     }
1034
1035     if ( ! mdoc_block_alloc(m, line, ppos, tok, arg))
1036         return(0);
1037
1038     head = body = NULL;
1039
1040     /*
1041     * Exception: Heads of 'It' macros in '-diag' lists are not
1042     * parsed, even though 'It' macros in general are parsed.
1043     */
1044     nparsed = MDOC_It == tok &&
1045         MDOC_Bl == m->last->parent->tok &&
1046         LIST_diag == m->last->parent->norm->Bl.type;

```

```

1048  /*
1049  * The 'Nd' macro has all arguments in its body: it's a hybrid
1050  * of block partial-explicit and full-implicit. Stupid.
1051  */

1053  if (MDOC_Nd == tok) {
1054      if (! mdoc_head_alloc(m, line, ppos, tok))
1055          return(0);
1056      head = m->last;
1057      if (! rew_sub(MDOC_HEAD, m, tok, line, ppos))
1058          return(0);
1059      if (! mdoc_body_alloc(m, line, ppos, tok))
1060          return(0);
1061      body = m->last;
1062  }

1064  ac = ARGS_ERROR;

1066  for ( ; ; ) {
1067      la = *pos;
1068      /* Initialise last-phrase-type with ARGS_PEND. */
1069      lac = ARGS_ERROR == ac ? ARGS_PEND : ac;
1070      ac = mdoc_args(m, line, pos, buf, tok, &p);

1072      if (ARGS_PUNCT == ac)
1073          break;

1075      if (ARGS_ERROR == ac)
1076          return(0);

1078      if (ARGS_EOLN == ac) {
1079          if (ARGS_PPHRASE != lac && ARGS_PHRASE != lac)
1080              break;
1081          /*
1082           * This is necessary: if the last token on a
1083           * line is a 'Ta' or tab, then we'll get
1084           * ARGS_EOLN, so we must be smart enough to
1085           * reopen our scope if the last parse was a
1086           * phrase or partial phrase.
1087           */
1088          if (! rew_sub(MDOC_BODY, m, tok, line, ppos))
1089              return(0);
1090          if (! mdoc_body_alloc(m, line, ppos, tok))
1091              return(0);
1092          body = m->last;
1093          break;
1094      }

1096      /*
1097       * Emit leading punctuation (i.e., punctuation before
1098       * the MDOC_HEAD) for non-phrase types.
1099       */

1101      if (NULL == head &&
1102          ARGS_PEND != ac &&
1103          ARGS_PHRASE != ac &&
1104          ARGS_PPHRASE != ac &&
1105          ARGS_QWORD != ac &&
1106          DELIM_OPEN == mdoc_isdelim(p)) {
1107          if (! dword(m, line, la, p, DELIM_OPEN))
1108              return(0);
1109          continue;
1110      }

1112      /* Open a head if one hasn't been opened. */

```

```

1114      if (NULL == head) {
1115          if (! mdoc_head_alloc(m, line, ppos, tok))
1116              return(0);
1117          head = m->last;
1118      }

1120      if (ARGS_PHRASE == ac ||
1121          ARGS_PEND == ac ||
1122          ARGS_PPHRASE == ac) {
1123          /*
1124           * If we haven't opened a body yet, rewind the
1125           * head; if we have, rewind that instead.
1126           */

1128          mtt = body ? MDOC_BODY : MDOC_HEAD;
1129          if (! rew_sub(mtt, m, tok, line, ppos))
1130              return(0);
1131          /* Then allocate our body context. */

1134          if (! mdoc_body_alloc(m, line, ppos, tok))
1135              return(0);
1136          body = m->last;

1138          /*
1139           * Process phrases: set whether we're in a
1140           * partial-phrase (this effects line handling)
1141           * then call down into the phrase parser.
1142           */

1144          if (ARGS_PPHRASE == ac)
1145              m->flags |= MDOC_PPHRASE;
1146          if (ARGS_PEND == ac && ARGS_PPHRASE == lac)
1147              m->flags |= MDOC_PPHRASE;

1149          if (! phrase(m, line, la, buf))
1150              return(0);

1152          m->flags &= ~MDOC_PPHRASE;
1153          continue;
1154      }

1156      ntok = nparsed || ARGS_QWORD == ac ?
1157          MDOC_MAX : lookup(tok, p);

1159      if (MDOC_MAX == ntok) {
1160          if (! dword(m, line, la, p, DELIM_MAX))
1161              return(0);
1162          continue;
1163      }

1165      if (! mdoc_macro(m, ntok, line, la, pos, buf))
1166          return(0);
1167      break;
1168  }

1170  if (NULL == head) {
1171      if (! mdoc_head_alloc(m, line, ppos, tok))
1172          return(0);
1173      head = m->last;
1174  }

1176  if (nl && ! append_delims(m, line, pos, buf))
1177      return(0);

```

```

1179      /* If we've already opened our body, exit now. */
1181      if (NULL != body)
1182          goto out;
1184      /*
1185       * If there is an open (i.e., unvalidated) sub-block requiring
1186       * explicit close-out, postpone switching the current block from
1187       * head to body until the rew_sub() call closing out that
1188       * sub-block.
1189       */
1190      for (n = m->last; n && n != head; n = n->parent) {
1191          if (MDOC_BLOCK == n->type &&
1192              MDOC_EXPLICIT & mdoc_macros[n->tok].flags &&
1193              ! (MDOC_VALID & n->flags)) {
1194              n->pending = head;
1195              return(1);
1196          }
1197      }
1199      /* Close out scopes to remain in a consistent state. */
1201      if ( ! rew_sub(MDOC_HEAD, m, tok, line, ppos))
1202          return(0);
1203      if ( ! mdoc_body_alloc(m, line, ppos, tok))
1204          return(0);
1206 out:
1207      if ( ! (MDOC_FREECOL & m->flags))
1208          return(1);
1210      if ( ! rew_sub(MDOC_BODY, m, tok, line, ppos))
1211          return(0);
1212      if ( ! rew_sub(MDOC_BLOCK, m, tok, line, ppos))
1213          return(0);
1215      m->flags &= ~MDOC_FREECOL;
1216      return(1);
1217 }
1220 static int
1221 blk_part_imp(MACRO_PROT_ARGS)
1222 {
1223     int            la, nl;
1224     enum mdoct     ntok;
1225     enum margserr  ac;
1226     char           *p;
1227     struct mdoc_node *blk; /* saved block context */
1228     struct mdoc_node *body; /* saved body context */
1229     struct mdoc_node *n;
1231     nl = MDOC_NEWLINE & m->flags;
1233     /*
1234      * A macro that spans to the end of the line. This is generally
1235      * (but not necessarily) called as the first macro. The block
1236      * has a head as the immediate child, which is always empty,
1237      * followed by zero or more opening punctuation nodes, then the
1238      * body (which may be empty, depending on the macro), then zero
1239      * or more closing punctuation nodes.
1240      */
1242     if ( ! mdoc_block_alloc(m, line, ppos, tok, NULL))
1243         return(0);

```

```

1245     blk = m->last;
1247     if ( ! mdoc_head_alloc(m, line, ppos, tok))
1248         return(0);
1249     if ( ! rew_sub(MDOC_HEAD, m, tok, line, ppos))
1250         return(0);
1252     /*
1253      * Open the body scope "on-demand", that is, after we've
1254      * processed all our the leading delimiters (open parenthesis,
1255      * etc.).
1256      */
1258     for (body = NULL; ; ) {
1259         la = *pos;
1260         ac = mdoc_args(m, line, pos, buf, tok, &p);
1262         if (ARGS_ERROR == ac)
1263             return(0);
1264         if (ARGS_EOLN == ac)
1265             break;
1266         if (ARGS_PUNCT == ac)
1267             break;
1269         if (NULL == body && ARGS_QWORD != ac &&
1270             DELIM_OPEN == mdoc_isdelim(p)) {
1271             if ( ! dword(m, line, la, p, DELIM_OPEN))
1272                 return(0);
1273             continue;
1274         }
1276         if (NULL == body) {
1277             if ( ! mdoc_body_alloc(m, line, ppos, tok))
1278                 return(0);
1279             body = m->last;
1280         }
1282         ntok = ARGS_QWORD == ac ? MDOC_MAX : lookup(tok, p);
1284         if (MDOC_MAX == ntok) {
1285             if ( ! dword(m, line, la, p, DELIM_MAX))
1286                 return(0);
1287             continue;
1288         }
1290         if ( ! mdoc_macro(m, ntok, line, la, pos, buf))
1291             return(0);
1292         break;
1293     }
1295     /* Clean-ups to leave in a consistent state. */
1297     if (NULL == body) {
1298         if ( ! mdoc_body_alloc(m, line, ppos, tok))
1299             return(0);
1300         body = m->last;
1301     }
1303     for (n = body->child; n && n->next; n = n->next)
1304         /* Do nothing. */ ;
1305
1306     /*
1307      * End of sentence spacing: if the last node is a text node and
1308      * has a trailing period, then mark it as being end-of-sentence.
1309      */

```

```

1311     if (n && MDOC_TEXT == n->type && n->string)
1312         if (mandoc_eos(n->string, strlen(n->string), 1))
1313             n->flags |= MDOC_EOS;

1315     /* Up-propagate the end-of-space flag. */

1317     if (n && (MDOC_EOS & n->flags)) {
1318         body->flags |= MDOC_EOS;
1319         body->parent->flags |= MDOC_EOS;
1320     }

1322     /*
1323      * If there is an open sub-block requiring explicit close-out,
1324      * postpone closing out the current block
1325      * until the rew_sub() call closing out the sub-block.
1326      */
1327     for (n = m->last; n && n != body && n != blk->parent; n = n->parent) {
1328         if (MDOC_BLOCK == n->type &&
1329             MDOC_EXPLICIT & mdoc_macros[n->tok].flags &&
1330             ! (MDOC_VALID & n->flags)) {
1331             make_pending(n, tok, m, line, ppos);
1332             if ( ! mdoc_endbody_alloc(m, line, ppos,
1333                 tok, body, ENDBODY_NOSPACE))
1334                 return(0);
1335             return(1);
1336         }
1337     }

1339     /*
1340      * If we can't rewind to our body, then our scope has already
1341      * been closed by another macro (like 'Oc' closing 'Op'). This
1342      * is ugly behaviour nodding its head to OpenBSD's overwhelming
1343      * cruddy use of 'Op' breakage.
1344      */
1345     if (n != body)
1346         mandoc_vmsg(MANDOCERR_SCOPENEST, m->parse, line, ppos,
1347             "%s broken", mdoc_macronames[tok]);

1349     if (n && ! rew_sub(MDOC_BODY, m, tok, line, ppos))
1350         return(0);

1352     /* Standard appending of delimiters. */

1354     if (nl && ! append_delims(m, line, pos, buf))
1355         return(0);

1357     /* Rewind scope, if applicable. */

1359     if (n && ! rew_sub(MDOC_BLOCK, m, tok, line, ppos))
1360         return(0);

1362     return(1);
1363 }

1366 static int
1367 blk_part_exp(MACRO_PROT_ARGS)
1368 {
1369     int             la, nl;
1370     enum margserr   ac;
1371     struct mdoc_node *head; /* keep track of head */
1372     struct mdoc_node *body; /* keep track of body */
1373     char            *p;
1374     enum mdoct      ntok;

1376     nl = MDOC_NEWLINE & m->flags;

```

```

1378     /*
1379      * The opening of an explicit macro having zero or more leading
1380      * punctuation nodes; a head with optional single element (the
1381      * case of 'Eo'); and a body that may be empty.
1382      */

1384     if ( ! mdoc_block_alloc(m, line, ppos, tok, NULL))
1385         return(0);

1387     for (head = body = NULL; ; ) {
1388         la = *pos;
1389         ac = mdoc_args(m, line, pos, buf, tok, &p);

1391         if (ARGS_ERROR == ac)
1392             return(0);
1393         if (ARGS_PUNCT == ac)
1394             break;
1395         if (ARGS_EOLIN == ac)
1396             break;

1398         /* Flush out leading punctuation. */

1400         if (NULL == head && ARGS_QWORD != ac &&
1401             DELIM_OPEN == mdoc_isdelim(p)) {
1402             assert(NULL == body);
1403             if ( ! dword(m, line, la, p, DELIM_OPEN))
1404                 return(0);
1405             continue;
1406         }

1408         if (NULL == head) {
1409             assert(NULL == body);
1410             if ( ! mdoc_head_alloc(m, line, ppos, tok))
1411                 return(0);
1412             head = m->last;
1413         }

1415         /*
1416          * 'Eo' gobbles any data into the head, but most other
1417          * macros just immediately close out and begin the body.
1418          */

1420         if (NULL == body) {
1421             assert(head);
1422             /* No check whether it's a macro! */
1423             if (MDOC_Eo == tok)
1424                 if ( ! dword(m, line, la, p, DELIM_MAX))
1425                     return(0);

1427             if ( ! rew_sub(MDOC_HEAD, m, tok, line, ppos))
1428                 return(0);
1429             if ( ! mdoc_body_alloc(m, line, ppos, tok))
1430                 return(0);
1431             body = m->last;

1433             if (MDOC_Eo == tok)
1434                 continue;
1435         }

1437         assert(NULL != head && NULL != body);

1439         ntok = ARGS_QWORD == ac ? MDOC_MAX : lookup(tok, p);

1441         if (MDOC_MAX == ntok) {
1442             if ( ! dword(m, line, la, p, DELIM_MAX))

```

```

1443         return(0);
1444         continue;
1445     }
1447     if ( ! mdoc_macro(m, ntok, line, la, pos, buf))
1448         return(0);
1449     break;
1450 }
1452 /* Clean-up to leave in a consistent state. */
1454 if (NULL == head)
1455     if ( ! mdoc_head_alloc(m, line, ppos, tok))
1456         return(0);
1458 if (NULL == body) {
1459     if ( ! rew_sub(MDOC_HEAD, m, tok, line, ppos))
1460         return(0);
1461     if ( ! mdoc_body_alloc(m, line, ppos, tok))
1462         return(0);
1463 }
1465 /* Standard appending of delimiters. */
1467 if ( ! nl)
1468     return(1);
1469 return(append_delims(m, line, pos, buf));
1470 }
1473 /* ARGSUSED */
1474 static int
1475 in_line_argn(MACRO_PROT_ARGS)
1476 {
1477     int             la, flushed, j, maxargs, nl;
1478     enum margserr   ac;
1479     enum margverr   av;
1480     struct mdoc_arg *arg;
1481     char            *p;
1482     enum mdoct      ntok;
1484     nl = MDOC_NEWLINE & m->flags;
1486     /*
1487     * A line macro that has a fixed number of arguments (maxargs).
1488     * Only open the scope once the first non-leading-punctuation is
1489     * found (unless MDOC_IGNDELIM is noted, like in 'Pf'), then
1490     * keep it open until the maximum number of arguments are
1491     * exhausted.
1492     */
1494     switch (tok) {
1495     case (MDOC_Ap):
1496         /* FALLTHROUGH */
1497     case (MDOC_No):
1498         /* FALLTHROUGH */
1499     case (MDOC_Ns):
1500         /* FALLTHROUGH */
1501     case (MDOC_Ux):
1502         maxargs = 0;
1503         break;
1504     case (MDOC_Bx):
1505         /* FALLTHROUGH */
1506     case (MDOC_Xr):
1507         maxargs = 2;
1508         break;

```

```

1509     default:
1510         maxargs = 1;
1511         break;
1512     }
1514     for (arg = NULL; ; ) {
1515         la = *pos;
1516         av = mdoc_argv(m, line, tok, &arg, pos, buf);
1518         if (ARGV_WORD == av) {
1519             *pos = la;
1520             break;
1521         }
1523         if (ARGV_EOLN == av)
1524             break;
1525         if (ARGV_ARG == av)
1526             continue;
1528         mdoc_argv_free(arg);
1529         return(0);
1530     }
1532     for (flushed = j = 0; ; ) {
1533         la = *pos;
1534         ac = mdoc_args(m, line, pos, buf, tok, &p);
1536         if (ARGS_ERROR == ac)
1537             return(0);
1538         if (ARGS_PUNCT == ac)
1539             break;
1540         if (ARGS_EOLN == ac)
1541             break;
1543         if ( ! (MDOC_IGNDELIM & mdoc_macros[tok].flags) &&
1544             ARGS_QWORD != ac && 0 == j &&
1545             DELIM_OPEN == mdoc_isdelim(p)) {
1546             if ( ! dword(m, line, la, p, DELIM_OPEN))
1547                 return(0);
1548             continue;
1549         } else if (0 == j)
1550             if ( ! mdoc_elem_alloc(m, line, la, tok, arg))
1551                 return(0);
1553         if (j == maxargs && ! flushed) {
1554             if ( ! rew_elem(m, tok))
1555                 return(0);
1556             flushed = 1;
1557         }
1559         ntok = ARGS_QWORD == ac ? MDOC_MAX : lookup(tok, p);
1561         if (MDOC_MAX != ntok) {
1562             if ( ! flushed && ! rew_elem(m, tok))
1563                 return(0);
1564             flushed = 1;
1565             if ( ! mdoc_macro(m, ntok, line, la, pos, buf))
1566                 return(0);
1567             j++;
1568             break;
1569         }
1571         if ( ! (MDOC_IGNDELIM & mdoc_macros[tok].flags) &&
1572             ARGS_QWORD != ac &&
1573             ! flushed &&
1574             DELIM_NONE != mdoc_isdelim(p)) {

```

```

1575         if ( ! rew_elem(m, tok))
1576             return(0);
1577         flushed = 1;
1578     }
1580     if ( ! dword(m, line, la, p, DELIM_MAX))
1581         return(0);
1582     j++;
1583 }
1585 if (0 == j && ! mdoc_elem_alloc(m, line, la, tok, arg))
1586     return(0);
1588 /* Close out in a consistent state. */
1590 if ( ! flushed && ! rew_elem(m, tok))
1591     return(0);
1592 if ( ! nl)
1593     return(1);
1594 return(append_delims(m, line, pos, buf));
1595 }

1598 static int
1599 in_line_eoln(MACRO_PROT_ARGS)
1600 {
1601     int          la;
1602     enum margserr ac;
1603     enum margverr av;
1604     struct mdoc_arg *arg;
1605     char         *p;
1606     enum mdoct   ntok;

1608     assert( ! (MDOC_PARSED & mdoc_macros[tok].flags));

1610     if (tok == MDOC_Pp)
1611         rew_sub(MDOC_BLOCK, m, MDOC_Nm, line, ppos);

1613     /* Parse macro arguments. */

1615     for (arg = NULL; ; ) {
1616         la = *pos;
1617         av = mdoc_argv(m, line, tok, &arg, pos, buf);

1619         if (ARGV_WORD == av) {
1620             *pos = la;
1621             break;
1622         }
1623         if (ARGV_EOLN == av)
1624             break;
1625         if (ARGV_ARG == av)
1626             continue;

1628         mdoc_argv_free(arg);
1629         return(0);
1630     }

1632     /* Open element scope. */

1634     if ( ! mdoc_elem_alloc(m, line, ppos, tok, arg))
1635         return(0);

1637     /* Parse argument terms. */

1639     for (;;) {
1640         la = *pos;

```

```

1641         ac = mdoc_args(m, line, pos, buf, tok, &p);

1643         if (ARGS_ERROR == ac)
1644             return(0);
1645         if (ARGS_EOLN == ac)
1646             break;

1648         ntok = ARGS_QWORD == ac ? MDOC_MAX : lookup(tok, p);

1650         if (MDOC_MAX == ntok) {
1651             if ( ! dword(m, line, la, p, DELIM_MAX))
1652                 return(0);
1653             continue;
1654         }

1656         if ( ! rew_elem(m, tok))
1657             return(0);
1658         return(mdoc_macro(m, ntok, line, la, pos, buf));
1659     }

1661     /* Close out (no delimiters). */

1663     return(rew_elem(m, tok));
1664 }

1667 /* ARGSUSED */
1668 static int
1669 ctx_synopsis(MACRO_PROT_ARGS)
1670 {
1671     int          nl;

1673     nl = MDOC_NEWLINE & m->flags;

1675     /* If we're not in the SYNOPSIS, go straight to in-line. */
1676     if ( ! (MDOC_SYNOPSIS & m->flags))
1677         return(in_line(m, tok, line, ppos, pos, buf));

1679     /* If we're a nested call, same place. */
1680     if ( ! nl)
1681         return(in_line(m, tok, line, ppos, pos, buf));

1683     /*
1684      * XXX: this will open a block scope; however, if later we end
1685      * up formatting the block scope, then child nodes will inherit
1686      * the formatting. Be careful.
1687      */
1688     if (MDOC_Nm == tok)
1689         return(blk_full(m, tok, line, ppos, pos, buf));
1690     assert(MDOC_Vt == tok);
1691     return(blk_part_imp(m, tok, line, ppos, pos, buf));
1692 }

1695 /* ARGSUSED */
1696 static int
1697 obsolete(MACRO_PROT_ARGS)
1698 {

1700     mdoc_pmsg(m, line, ppos, MANDOCERR_MACROOBS);
1701     return(1);
1702 }

1705 /*
1706  * Phrases occur within 'B1 -column' entries, separated by 'Ta' or tabs.

```

```

1707 * They're unusual because they're basically free-form text until a
1708 * macro is encountered.
1709 */
1710 static int
1711 phrase(struct mdoc *m, int line, int ppos, char *buf)
1712 {
1713     int            la, pos;
1714     enum margserr  ac;
1715     enum mdoct     ntok;
1716     char           *p;
1717
1718     for (pos = ppos; ; ) {
1719         la = pos;
1720
1721         ac = mdoc_zargs(m, line, &pos, buf, &p);
1722
1723         if (ARGS_ERROR == ac)
1724             return(0);
1725         if (ARGS_EOLN == ac)
1726             break;
1727
1728         ntok = ARGS_QWORD == ac ? MDOC_MAX : lookup_raw(p);
1729
1730         if (MDOC_MAX == ntok) {
1731             if ( ! dword(m, line, la, p, DELIM_MAX))
1732                 return(0);
1733             continue;
1734         }
1735
1736         if ( ! mdoc_macro(m, ntok, line, la, &pos, buf))
1737             return(0);
1738         return(append_delims(m, line, &pos, buf));
1739     }
1740
1741     return(1);
1742 }
1743
1744
1745 /* ARGSUSED */
1746 static int
1747 phrase_ta(MACRO_PROT_ARGS)
1748 {
1749     int            la;
1750     enum mdoct     ntok;
1751     enum margserr  ac;
1752     char           *p;
1753
1754     /*
1755      * FIXME: this is overly restrictive: if the 'Ta' is unexpected,
1756      * it should simply error out with ARGSLOST.
1757      */
1758
1759     if ( ! rew_sub(MDOC_BODY, m, MDOC_It, line, ppos))
1760         return(0);
1761     if ( ! mdoc_body_alloc(m, line, ppos, MDOC_It))
1762         return(0);
1763
1764     for (;;) {
1765         la = *ppos;
1766         ac = mdoc_zargs(m, line, pos, buf, &p);
1767
1768         if (ARGS_ERROR == ac)
1769             return(0);
1770         if (ARGS_EOLN == ac)
1771             break;

```

```

1773         ntok = ARGS_QWORD == ac ? MDOC_MAX : lookup_raw(p);
1774
1775         if (MDOC_MAX == ntok) {
1776             if ( ! dword(m, line, la, p, DELIM_MAX))
1777                 return(0);
1778             continue;
1779         }
1780
1781         if ( ! mdoc_macro(m, ntok, line, la, pos, buf))
1782             return(0);
1783         return(append_delims(m, line, pos, buf));
1784     }
1785
1786     return(1);
1787 }

```

```

*****
15647 Sat Jul 19 19:24:31 2014
new/usr/src/cmd/mandoc/mdoc_man.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: mdoc_man.c,v 1.9 2011/10/24 21:47:59 schwarze Exp $ */
2 /*
3  * Copyright (c) 2011 Ingo Schwarze <schwarze@openbsd.org>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <stdio.h>
22 #include <string.h>
23
24 #include "mandoc.h"
25 #include "man.h"
26 #include "mdoc.h"
27 #include "main.h"
28
29 #define DECL_ARGS const struct mdoc_meta *m, \
30                  const struct mdoc_node *n, \
31                  struct mman *mm
32
33 struct mman {
34     int      need_space; /* next word needs prior ws */
35     int      need_nl; /* next word needs prior nl */
36 };
37
38 struct manact {
39     int      (*cond)(DECL_ARGS); /* DON'T run actions */
40     int      (*pre)(DECL_ARGS); /* pre-node action */
41     void     (*post)(DECL_ARGS); /* post-node action */
42     const char *prefix; /* pre-node string constant */
43     const char *suffix; /* post-node string constant */
44 };
45
46 static int      cond_body(DECL_ARGS);
47 static int      cond_head(DECL_ARGS);
48 static void     post_bd(DECL_ARGS);
49 static void     post_dl(DECL_ARGS);
50 static void     post_enc(DECL_ARGS);
51 static void     post_nm(DECL_ARGS);
52 static void     post_percent(DECL_ARGS);
53 static void     post_pf(DECL_ARGS);
54 static void     post_sect(DECL_ARGS);
55 static void     post_sp(DECL_ARGS);
56 static int      pre_ap(DECL_ARGS);

```

```

57 static int      pre_bd(DECL_ARGS);
58 static int      pre_br(DECL_ARGS);
59 static int      pre_bx(DECL_ARGS);
60 static int      pre_dl(DECL_ARGS);
61 static int      pre_enc(DECL_ARGS);
62 static int      pre_it(DECL_ARGS);
63 static int      pre_nm(DECL_ARGS);
64 static int      pre_ns(DECL_ARGS);
65 static int      pre_pp(DECL_ARGS);
66 static int      pre_sp(DECL_ARGS);
67 static int      pre_sect(DECL_ARGS);
68 static int      pre_ux(DECL_ARGS);
69 static int      pre_xr(DECL_ARGS);
70 static void     print_word(struct mman *, const char *);
71 static void     print_node(DECL_ARGS);
72
73 static const struct manact manacts[MDOC_MAX + 1] = {
74     { NULL, pre_ap, NULL, NULL, NULL }, /* Ap */
75     { NULL, NULL, NULL, NULL, NULL }, /* Dd */
76     { NULL, NULL, NULL, NULL, NULL }, /* Dt */
77     { NULL, NULL, NULL, NULL, NULL }, /* Os */
78     { NULL, pre_sect, post_sect, ".SH", NULL }, /* Sh */
79     { NULL, pre_sect, post_sect, ".SS", NULL }, /* Ss */
80     { NULL, pre_pp, NULL, NULL, NULL }, /* Pp */
81     { cond_body, pre_dl, post_dl, NULL, NULL }, /* Dl */
82     { cond_body, pre_dl, post_dl, NULL, NULL }, /* D1 */
83     { cond_body, pre_bd, post_bd, NULL, NULL }, /* Bd */
84     { NULL, NULL, NULL, NULL, NULL }, /* Ed */
85     { NULL, NULL, NULL, NULL, NULL }, /* Bl */
86     { NULL, NULL, NULL, NULL, NULL }, /* El */
87     { NULL, pre_it, NULL, NULL, NULL }, /* It */
88     { NULL, pre_enc, post_enc, "\\fI", "\\fP" }, /* Ad */
89     { NULL, NULL, NULL, NULL, NULL }, /* An */
90     { NULL, pre_enc, post_enc, "\\fI", "\\fP" }, /* Ar */
91     { NULL, pre_enc, post_enc, "\\fB", "\\fP" }, /* Cd */
92     { NULL, pre_enc, post_enc, "\\fB", "\\fP" }, /* Cm */
93     { NULL, pre_enc, post_enc, "\\fR", "\\fP" }, /* Dv */
94     { NULL, pre_enc, post_enc, "\\fR", "\\fP" }, /* Er */
95     { NULL, pre_enc, post_enc, "\\fR", "\\fP" }, /* Ev */
96     { NULL, pre_enc, post_enc, "The \\fB",
97       "\\fP\nutility exits 0 on success, and >0 if an error occurs."
98     }, /* Ex */
99     { NULL, NULL, NULL, NULL, NULL }, /* Fa */
100    { NULL, NULL, NULL, NULL, NULL }, /* Fd */
101    { NULL, pre_enc, post_enc, "\\fB-", "\\fP" }, /* Fl */
102    { NULL, NULL, NULL, NULL, NULL }, /* Fn */
103    { NULL, NULL, NULL, NULL, NULL }, /* Ft */
104    { NULL, pre_enc, post_enc, "\\fB", "\\fP" }, /* Ic */
105    { NULL, NULL, NULL, NULL, NULL }, /* In */
106    { NULL, pre_enc, post_enc, "\\fR", "\\fP" }, /* Li */
107    { cond_head, pre_enc, NULL, "\\- ", NULL }, /* Nd */
108    { NULL, pre_nm, post_nm, NULL, NULL }, /* Nm */
109    { cond_body, pre_enc, post_enc, "[", "]" }, /* Op */
110    { NULL, NULL, NULL, NULL, NULL }, /* Ot */
111    { NULL, pre_enc, post_enc, "\\fI", "\\fP" }, /* Pa */
112    { NULL, pre_enc, post_enc, "The \\fB",
113      "\\fP\nfunction returns the value 0 if successful;\n"
114      "otherwise the value -1 is returned and the global\n"
115      "variable \\fIerrno\\fP is set to indicate the error."
116    }, /* Rv */
117    { NULL, NULL, NULL, NULL, NULL }, /* St */
118    { NULL, NULL, NULL, NULL, NULL }, /* Va */
119    { NULL, NULL, NULL, NULL, NULL }, /* Vt */
120    { NULL, pre_xr, NULL, NULL, NULL }, /* Xr */
121    { NULL, NULL, post_percent, NULL, NULL }, /* %A */
122    { NULL, NULL, NULL, NULL, NULL }, /* %B */

```



```

123 NULL, NULL, post_percent, NULL, NULL }, /* _%D */
124 NULL, NULL, NULL, NULL, NULL }, /* %I */
125 NULL, pre_enc, post_percent, "\\fi", "\\fP" }, /* %J */
126 NULL, NULL, NULL, NULL, NULL }, /* %N */
127 NULL, NULL, NULL, NULL, NULL }, /* %O */
128 NULL, NULL, NULL, NULL, NULL }, /* %P */
129 NULL, NULL, NULL, NULL, NULL }, /* %R */
130 NULL, pre_enc, post_percent, "", "" }, /* %T */
131 NULL, NULL, NULL, NULL, NULL }, /* %V */
132 NULL, NULL, NULL, NULL, NULL }, /* Ac */
133 cond_body, pre_enc, post_enc, "<", ">" }, /* Ao */
134 cond_body, pre_enc, post_enc, "<", ">" }, /* Aq */
135 NULL, NULL, NULL, NULL, NULL }, /* At */
136 NULL, NULL, NULL, NULL, NULL }, /* Bc */
137 NULL, NULL, NULL, NULL, NULL }, /* Bf */
138 cond_body, pre_enc, post_enc, "[", "]" }, /* Bo */
139 cond_body, pre_enc, post_enc, "[", "]" }, /* Bq */
140 NULL, pre_ux, NULL, "BSD/OS", NULL }, /* Bsx */
141 NULL, pre_bx, NULL, NULL, NULL }, /* Bx */
142 NULL, NULL, NULL, NULL, NULL }, /* Db */
143 NULL, NULL, NULL, NULL, NULL }, /* Dc */
144 cond_body, pre_enc, post_enc, "\"", "\"" }, /* Do */
145 cond_body, pre_enc, post_enc, "\"", "\"" }, /* Dq */
146 NULL, NULL, NULL, NULL, NULL }, /* Ec */
147 NULL, NULL, NULL, NULL, NULL }, /* EF */
148 NULL, pre_enc, post_enc, "\\fi", "\\fP" }, /* Em */
149 NULL, NULL, NULL, NULL, NULL }, /* Eo */
150 NULL, pre_ux, NULL, "FreeBSD", NULL }, /* Fx */
151 NULL, pre_enc, post_enc, "\\fB", "\\fP" }, /* Ms */
152 NULL, NULL, NULL, NULL, NULL }, /* No */
153 NULL, pre_ns, NULL, NULL, NULL }, /* Ns */
154 NULL, pre_ux, NULL, "NetBSD", NULL }, /* Nx */
155 NULL, pre_ux, NULL, "OpenBSD", NULL }, /* Ox */
156 NULL, NULL, NULL, NULL, NULL }, /* Pc */
157 NULL, NULL, post_pf, NULL, NULL }, /* Pf */
158 cond_body, pre_enc, post_enc, "(", ")" }, /* Po */
159 cond_body, pre_enc, post_enc, "(", ")" }, /* Pq */
160 NULL, NULL, NULL, NULL, NULL }, /* Qc */
161 cond_body, pre_enc, post_enc, "\"", "\"" }, /* Ql */
162 cond_body, pre_enc, post_enc, "\"", "\"" }, /* Qo */
163 cond_body, pre_enc, post_enc, "\"", "\"" }, /* Qq */
164 NULL, NULL, NULL, NULL, NULL }, /* Re */
165 cond_body, pre_pp, NULL, NULL, NULL }, /* Rs */
166 NULL, NULL, NULL, NULL, NULL }, /* Sc */
167 cond_body, pre_enc, post_enc, "\"", "\"" }, /* So */
168 cond_body, pre_enc, post_enc, "\"", "\"" }, /* Sq */
169 NULL, NULL, NULL, NULL, NULL }, /* Sm */
170 NULL, pre_enc, post_enc, "\\fi", "\\fP" }, /* Sx */
171 NULL, pre_enc, post_enc, "\\fB", "\\fP" }, /* Sy */
172 NULL, pre_enc, post_enc, "\\fR", "\\fP" }, /* Tn */
173 NULL, pre_ux, NULL, "UNIX", NULL }, /* Ux */
174 NULL, NULL, NULL, NULL, NULL }, /* Xc */
175 NULL, NULL, NULL, NULL, NULL }, /* Xo */
176 NULL, NULL, NULL, NULL, NULL }, /* Fo */
177 NULL, NULL, NULL, NULL, NULL }, /* Fc */
178 cond_body, pre_enc, post_enc, "[", "]" }, /* Oo */
179 NULL, NULL, NULL, NULL, NULL }, /* Oc */
180 NULL, NULL, NULL, NULL, NULL }, /* Bk */
181 NULL, NULL, NULL, NULL, NULL }, /* Ek */
182 NULL, pre_ux, NULL, "is currently in beta test.", NULL }, /* Bt */
183 NULL, NULL, NULL, NULL, NULL }, /* Hf */
184 NULL, NULL, NULL, NULL, NULL }, /* Fr */
185 NULL, pre_ux, NULL, "currently under development.", NULL }, /* Ud */
186 NULL, NULL, NULL, NULL, NULL }, /* Lb */
187 NULL, pre_pp, NULL, NULL, NULL }, /* Lp */
188 NULL, NULL, NULL, NULL, NULL }, /* Lk */

```

```

189 NULL, NULL, NULL, NULL, NULL }, /* Mt */
190 cond_body, pre_enc, post_enc, "{", "}" }, /* Brq */
191 cond_body, pre_enc, post_enc, "{", "}" }, /* Bro */
192 NULL, NULL, NULL, NULL, NULL }, /* Brc */
193 NULL, NULL, NULL, NULL, NULL }, /* %C */
194 NULL, NULL, NULL, NULL, NULL }, /* _Es */
195 NULL, NULL, NULL, NULL, NULL }, /* _En */
196 NULL, pre_ux, NULL, "DragonFly", NULL }, /* Dx */
197 NULL, NULL, NULL, NULL, NULL }, /* %Q */
198 NULL, pre_br, NULL, NULL, NULL }, /* br */
199 NULL, pre_sp, post_sp, NULL, NULL }, /* sp */
200 NULL, NULL, NULL, NULL, NULL }, /* %U */
201 NULL, NULL, NULL, NULL, NULL }, /* Ta */
202 NULL, NULL, NULL, NULL, NULL }, /* ROOT */
203 };

204 static void
205 print_word(struct mman *mm, const char *s)
206 {
209     if (mm->need_nl) {
210         /*
211          * If we need a newline, print it now and start afresh.
212          */
213         putchar('\n');
214         mm->need_space = 0;
215         mm->need_nl = 0;
216     } else if (mm->need_space && '\0' != s[0])
217         /*
218          * If we need a space, only print it before
219          * (1) a nonzero length word;
220          * (2) a word that is non-punctuation; and
221          * (3) if punctuation, non-terminating punctuation.
222          */
223         if (NULL == strchr(".,;]?!", s[0]) || '\0' != s[1])
224             putchar(' ');

226     /*
227      * Reassign needing space if we're not following opening
228      * punctuation.
229      */
230     mm->need_space =
231         ((' ' != s[0] && '[' != s[0]) || '\0' != s[1]);

233     for ( ; *s; s++) {
234         switch (*s) {
235             case (ASCII_NBRSP):
236                 printf("\\~");
237                 break;
238             case (ASCII_HYPH):
239                 putchar('-');
240                 break;
241             default:
242                 putchar((unsigned char)*s);
243                 break;
244         }
245     }
246 }

248 void
249 man_man(void *arg, const struct man *man)
250 {
252     /*
253      * Dump the keep buffer.
254      * We're guaranteed by now that this exists (is non-NULL).

```

```

255     * Flush stdout afterward, just in case.
256     */
257     fputs(mparse_getkeep(man_mparse(man)), stdout);
258     fflush(stdout);
259 }

261 void
262 man_mdoc(void *arg, const struct mdoc *mdoc)
263 {
264     const struct mdoc_meta *m;
265     const struct mdoc_node *n;
266     struct mman          mm;

268     m = mdoc_meta(mdoc);
269     n = mdoc_node(mdoc);

271     printf(".TH \"%s\" \"%s\" \"%s\" \"%s\" \"%s\"",
272           m->title, m->msec, m->date, m->os, m->vol);

274     memset(&mm, 0, sizeof(struct mman));

276     mm.need_nl = 1;
277     print_node(m, n, &mm);
278     putchar('\n');
279 }

281 static void
282 print_node(DECL_ARGS)
283 {
284     const struct mdoc_node *prev, *sub;
285     const struct manact   *act;
286     int                    cond, do_sub;
287
288     /*
289     * Break the line if we were parsed subsequent the current node.
290     * This makes the page structure be more consistent.
291     */
292     prev = n->prev ? n->prev : n->parent;
293     if (prev && prev->line < n->line)
294         mm->need_nl = 1;

296     act = NULL;
297     cond = 0;
298     do_sub = 1;

300     if (MDOC_TEXT == n->type) {
301         /*
302         * Make sure that we don't happen to start with a
303         * control character at the start of a line.
304         */
305         if (mm->need_nl && ('.' == *n->string ||
306                             '\\"' == *n->string)) {
307             print_word(mm, "\\&");
308             mm->need_space = 0;
309         }
310         print_word(mm, n->string);
311     } else {
312         /*
313         * Conditionally run the pre-node action handler for a
314         * node.
315         */
316         act = manacts + n->tok;
317         cond = NULL == act->cond || (*act->cond)(m, n, mm);
318         if (cond && act->pre)
319             do_sub = (*act->pre)(m, n, mm);
320     }

```

```

322     /*
323     * Conditionally run all child nodes.
324     * Note that this iterates over children instead of using
325     * recursion. This prevents unnecessary depth in the stack.
326     */
327     if (do_sub)
328         for (sub = n->child; sub; sub = sub->next)
329             print_node(m, sub, mm);

331     /*
332     * Lastly, conditionally run the post-node handler.
333     */
334     if (cond && act->post)
335         (*act->post)(m, n, mm);
336 }

338 static int
339 cond_head(DECL_ARGS)
340 {
342     return(MDOC_HEAD == n->type);
343 }

345 static int
346 cond_body(DECL_ARGS)
347 {
349     return(MDOC_BODY == n->type);
350 }

352 /*
353 * Output a font encoding before a node, e.g., \fR.
354 * This obviously has no trailing space.
355 */
356 static int
357 pre_enc(DECL_ARGS)
358 {
359     const char *prefix;

361     prefix = manacts[n->tok].prefix;
362     if (NULL == prefix)
363         return(1);
364     print_word(mm, prefix);
365     mm->need_space = 0;
366     return(1);
367 }

369 /*
370 * Output a font encoding subsequent a node, e.g., \fP.
371 */
372 static void
373 post_enc(DECL_ARGS)
374 {
375     const char *suffix;

377     suffix = manacts[n->tok].suffix;
378     if (NULL == suffix)
379         return;
380     mm->need_space = 0;
381     print_word(mm, suffix);
382 }

384 /*
385 * Used in listings (percent = %A, e.g.).
386 * FIXME: this is incomplete.

```

```

387 * It doesn't print a nice ", and" for lists.
388 */
389 static void
390 post_percent(DECL_ARGS)
391 {
392
393     post_enc(m, n, mm);
394     if (n->next)
395         print_word(mm, ",");
396     else {
397         print_word(mm, ".");
398         mm->need_nl = 1;
399     }
400 }
401
402 /*
403 * Print before a section header.
404 */
405 static int
406 pre_sect(DECL_ARGS)
407 {
408
409     if (MDOC_HEAD != n->type)
410         return(1);
411     mm->need_nl = 1;
412     print_word(mm, manacts[n->tok].prefix);
413     print_word(mm, "\\");
414     mm->need_space = 0;
415     return(1);
416 }
417
418 /*
419 * Print subsequent a section header.
420 */
421 static void
422 post_sect(DECL_ARGS)
423 {
424
425     if (MDOC_HEAD != n->type)
426         return;
427     mm->need_space = 0;
428     print_word(mm, "\\");
429     mm->need_nl = 1;
430 }
431
432 static int
433 pre_ap(DECL_ARGS)
434 {
435
436     mm->need_space = 0;
437     print_word(mm, "'");
438     mm->need_space = 0;
439     return(0);
440 }
441
442 static int
443 pre_bd(DECL_ARGS)
444 {
445
446     if (DISP_unfilled == n->norm->Bd.type ||
447         DISP_literal == n->norm->Bd.type) {
448         mm->need_nl = 1;
449         print_word(mm, ".nf");
450     }
451     mm->need_nl = 1;
452     return(1);

```

```

453 }
454
455 static void
456 post_bd(DECL_ARGS)
457 {
458
459     if (DISP_unfilled == n->norm->Bd.type ||
460         DISP_literal == n->norm->Bd.type) {
461         mm->need_nl = 1;
462         print_word(mm, ".fi");
463     }
464     mm->need_nl = 1;
465 }
466
467 static int
468 pre_br(DECL_ARGS)
469 {
470
471     mm->need_nl = 1;
472     print_word(mm, ".br");
473     mm->need_nl = 1;
474     return(0);
475 }
476
477 static int
478 pre_bx(DECL_ARGS)
479 {
480
481     n = n->child;
482     if (n) {
483         print_word(mm, n->string);
484         mm->need_space = 0;
485         n = n->next;
486     }
487     print_word(mm, "BSD");
488     if (NULL == n)
489         return(0);
490     mm->need_space = 0;
491     print_word(mm, "-");
492     mm->need_space = 0;
493     print_word(mm, n->string);
494     return(0);
495 }
496
497 static int
498 pre_dl(DECL_ARGS)
499 {
500
501     mm->need_nl = 1;
502     print_word(mm, ".RS 6n");
503     mm->need_nl = 1;
504     return(1);
505 }
506
507 static void
508 post_dl(DECL_ARGS)
509 {
510
511     mm->need_nl = 1;
512     print_word(mm, ".RE");
513     mm->need_nl = 1;
514 }
515
516 static int
517 pre_it(DECL_ARGS)
518 {

```

```

519     const struct mdoc_node *bln;

521     if (MDOC_HEAD == n->type) {
522         mm->need_nl = 1;
523         print_word(mm, ".TP");
524         bln = n->parent->parent->prev;
525         switch (bln->norm->Bl.type) {
526             case (LIST_bullet):
527                 print_word(mm, "4n");
528                 mm->need_nl = 1;
529                 print_word(mm, "\\fBo\\fP");
530                 break;
531             default:
532                 if (bln->norm->Bl.width)
533                     print_word(mm, bln->norm->Bl.width);
534                 break;
535         }
536         mm->need_nl = 1;
537     }
538     return(1);
539 }

541 static int
542 pre_nm(DECL_ARGS)
543 {
544
545     if (MDOC_ELEM != n->type && MDOC_HEAD != n->type)
546         return(1);
547     print_word(mm, "\\fB");
548     mm->need_space = 0;
549     if (NULL == n->child)
550         print_word(mm, m->name);
551     return(1);
552 }

554 static void
555 post_nm(DECL_ARGS)
556 {
557
558     if (MDOC_ELEM != n->type && MDOC_HEAD != n->type)
559         return;
560     mm->need_space = 0;
561     print_word(mm, "\\fP");
562 }

564 static int
565 pre_ns(DECL_ARGS)
566 {
567
568     mm->need_space = 0;
569     return(0);
570 }

572 static void
573 post_pf(DECL_ARGS)
574 {
575
576     mm->need_space = 0;
577 }

579 static int
580 pre_pp(DECL_ARGS)
581 {
582
583     mm->need_nl = 1;
584     if (MDOC_It == n->parent->tok)

```

```

585         print_word(mm, ".sp");
586     else
587         print_word(mm, ".PP");
588     mm->need_nl = 1;
589     return(1);
590 }

592 static int
593 pre_sp(DECL_ARGS)
594 {
595
596     mm->need_nl = 1;
597     print_word(mm, ".sp");
598     return(1);
599 }

601 static void
602 post_sp(DECL_ARGS)
603 {
604
605     mm->need_nl = 1;
606 }

608 static int
609 pre_xr(DECL_ARGS)
610 {
611
612     n = n->child;
613     if (NULL == n)
614         return(0);
615     print_node(m, n, mm);
616     n = n->next;
617     if (NULL == n)
618         return(0);
619     mm->need_space = 0;
620     print_word(mm, "(");
621     print_node(m, n, mm);
622     print_word(mm, ")");
623     return(0);
624 }

626 static int
627 pre_ux(DECL_ARGS)
628 {
629
630     print_word(mm, manacts[n->tok].prefix);
631     if (NULL == n->child)
632         return(0);
633     mm->need_space = 0;
634     print_word(mm, "\\~");
635     mm->need_space = 0;
636     return(1);
637 }

```

```

*****
44751 Sat Jul 19 19:24:31 2014
new/usr/src/cmd/mandoc/mdoc_term.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: mdoc_term.c,v 1.238 2011/11/13 13:15:14 schwarze Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2010 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <sys/types.h>
23
24 #include <assert.h>
25 #include <ctype.h>
26 #include <stdint.h>
27 #include <stdio.h>
28 #include <stdlib.h>
29 #include <string.h>
30
31 #include "mandoc.h"
32 #include "out.h"
33 #include "term.h"
34 #include "mdoc.h"
35 #include "main.h"
36
37 struct termpair {
38     struct termpair *ppair;
39     int count;
40 };
41
42 #define DECL_ARGS struct termp *p, \
43                  struct termpair *pair, \
44                  const struct mdoc_meta *m, \
45                  const struct mdoc_node *n
46
47 struct termact {
48     int (*pre)(DECL_ARGS);
49     void (*post)(DECL_ARGS);
50 };
51
52 static size_t a2width(const struct termp *, const char *);
53 static size_t a2height(const struct termp *, const char *);
54 static size_t a2offs(const struct termp *, const char *);
55
56 static void print_bvspace(struct termp *,

```

```

57     const struct mdoc_node *,
58     const struct mdoc_node *);
59 static void print_mdoc_node(DECL_ARGS);
60 static void print_mdoc_nodelist(DECL_ARGS);
61 static void print_mdoc_head(struct termp *, const void *);
62 static void print_mdoc_foot(struct termp *, const void *);
63 static void synopsis_pre(struct termp *,
64     const struct mdoc_node *);
65
66 static void term__post(DECL_ARGS);
67 static void term_t_post(DECL_ARGS);
68 static void term_an_post(DECL_ARGS);
69 static void term_bd_post(DECL_ARGS);
70 static void term_bk_post(DECL_ARGS);
71 static void term_bl_post(DECL_ARGS);
72 static void term_dl_post(DECL_ARGS);
73 static void term_fo_post(DECL_ARGS);
74 static void term_in_post(DECL_ARGS);
75 static void term_it_post(DECL_ARGS);
76 static void term_lb_post(DECL_ARGS);
77 static void term_nm_post(DECL_ARGS);
78 static void term_pf_post(DECL_ARGS);
79 static void term_quote_post(DECL_ARGS);
80 static void term_sh_post(DECL_ARGS);
81 static void term_ss_post(DECL_ARGS);
82
83 static int term_a_pre(DECL_ARGS);
84 static int term_t_pre(DECL_ARGS);
85 static int term_an_pre(DECL_ARGS);
86 static int term_ap_pre(DECL_ARGS);
87 static int term_bd_pre(DECL_ARGS);
88 static int term_bf_pre(DECL_ARGS);
89 static int term_bk_pre(DECL_ARGS);
90 static int term_bl_pre(DECL_ARGS);
91 static int term_bold_pre(DECL_ARGS);
92 static int term_bt_pre(DECL_ARGS);
93 static int term_bx_pre(DECL_ARGS);
94 static int term_cd_pre(DECL_ARGS);
95 static int term_dl_pre(DECL_ARGS);
96 static int term_ex_pre(DECL_ARGS);
97 static int term_fa_pre(DECL_ARGS);
98 static int term_fd_pre(DECL_ARGS);
99 static int term_fl_pre(DECL_ARGS);
100 static int term_fn_pre(DECL_ARGS);
101 static int term_fo_pre(DECL_ARGS);
102 static int term_ft_pre(DECL_ARGS);
103 static int term_igndelim_pre(DECL_ARGS);
104 static int term_in_pre(DECL_ARGS);
105 static int term_it_pre(DECL_ARGS);
106 static int term_li_pre(DECL_ARGS);
107 static int term_lk_pre(DECL_ARGS);
108 static int term_nd_pre(DECL_ARGS);
109 static int term_nm_pre(DECL_ARGS);
110 static int term_ns_pre(DECL_ARGS);
111 static int term_quote_pre(DECL_ARGS);
112 static int term_rs_pre(DECL_ARGS);
113 static int term_rv_pre(DECL_ARGS);
114 static int term_sh_pre(DECL_ARGS);
115 static int term_sm_pre(DECL_ARGS);
116 static int term_sp_pre(DECL_ARGS);
117 static int term_ss_pre(DECL_ARGS);
118 static int term_under_pre(DECL_ARGS);
119 static int term_ud_pre(DECL_ARGS);
120 static int term_vt_pre(DECL_ARGS);
121 static int term_xr_pre(DECL_ARGS);
122 static int term_xx_pre(DECL_ARGS);

```

```

124 static const struct termact termacts[MDOC_MAX] = {
125     term_ap_pre, NULL, /* Ap */
126     NULL, NULL, /* Dd */
127     NULL, NULL, /* Dt */
128     NULL, NULL, /* Os */
129     term_sh_pre, term_sh_post, /* Sh */
130     term_ss_pre, term_ss_post, /* Ss */
131     term_sp_pre, NULL, /* Pp */
132     term_dl_pre, term_dl_post, /* D1 */
133     term_d1_pre, term_d1_post, /* D1 */
134     term_bd_pre, term_bd_post, /* Bd */
135     NULL, NULL, /* Ed */
136     term_bl_pre, term_bl_post, /* B1 */
137     NULL, NULL, /* E1 */
138     term_it_pre, term_it_post, /* It */
139     term_under_pre, NULL, /* Ad */
140     term_an_pre, term_an_post, /* An */
141     term_under_pre, NULL, /* Ar */
142     term_cd_pre, NULL, /* Cd */
143     term_bold_pre, NULL, /* Cm */
144     NULL, NULL, /* Dv */
145     NULL, NULL, /* Er */
146     NULL, NULL, /* Ev */
147     term_ex_pre, NULL, /* Ex */
148     term_fa_pre, NULL, /* Fa */
149     term_fd_pre, NULL, /* Fd */
150     term_fl_pre, NULL, /* Fl */
151     term_fn_pre, NULL, /* Fn */
152     term_ft_pre, NULL, /* Ft */
153     term_bold_pre, NULL, /* Ic */
154     term_in_pre, term_in_post, /* In */
155     term_li_pre, NULL, /* Li */
156     term_nd_pre, NULL, /* Nd */
157     term_nm_pre, term_nm_post, /* Nm */
158     term_quote_pre, term_quote_post, /* Op */
159     NULL, NULL, /* Ot */
160     term_under_pre, NULL, /* Pa */
161     term_rv_pre, NULL, /* Rv */
162     NULL, NULL, /* St */
163     term_under_pre, NULL, /* Va */
164     term_vt_pre, NULL, /* Vt */
165     term_xr_pre, NULL, /* Xr */
166     term_a_pre, term_post, /* %A */
167     term_under_pre, term_post, /* %B */
168     NULL, term_post, /* %D */
169     term_under_pre, term_post, /* %I */
170     term_under_pre, term_post, /* %J */
171     NULL, term_post, /* %N */
172     NULL, term_post, /* %O */
173     NULL, term_post, /* %P */
174     NULL, term_post, /* %R */
175     term_t_pre, term_t_post, /* %T */
176     NULL, term_post, /* %V */
177     NULL, NULL, /* Ac */
178     term_quote_pre, term_quote_post, /* Ao */
179     term_quote_pre, term_quote_post, /* Aq */
180     NULL, NULL, /* At */
181     NULL, NULL, /* Bc */
182     term_bf_pre, NULL, /* Bf */
183     term_quote_pre, term_quote_post, /* Bo */
184     term_quote_pre, term_quote_post, /* Bq */
185     term_xx_pre, NULL, /* Bsx */
186     term_bx_pre, NULL, /* Bx */
187     NULL, NULL, /* Db */
188     NULL, NULL, /* Dc */

```

```

189     term_quote_pre, term_quote_post, /* Do */
190     term_quote_pre, term_quote_post, /* Dq */
191     NULL, NULL, /* Ec */ /* FIXME: no space */
192     NULL, NULL, /* Ef */
193     term_under_pre, NULL, /* Em */
194     term_quote_pre, term_quote_post, /* Eo */
195     term_xx_pre, NULL, /* Fx */
196     term_bold_pre, NULL, /* Ms */
197     term_igndelim_pre, NULL, /* No */
198     term_ns_pre, NULL, /* Ns */
199     term_xx_pre, NULL, /* Nx */
200     term_xx_pre, NULL, /* Ox */
201     NULL, NULL, /* Pc */
202     term_igndelim_pre, term_pf_post, /* Pf */
203     term_quote_pre, term_quote_post, /* Po */
204     term_quote_pre, term_quote_post, /* Pq */
205     NULL, NULL, /* Qc */
206     term_quote_pre, term_quote_post, /* Ql */
207     term_quote_pre, term_quote_post, /* Qo */
208     term_quote_pre, term_quote_post, /* Qq */
209     NULL, NULL, /* Re */
210     term_rs_pre, NULL, /* Rs */
211     NULL, NULL, /* Sc */
212     term_quote_pre, term_quote_post, /* So */
213     term_quote_pre, term_quote_post, /* Sq */
214     term_sm_pre, NULL, /* Sm */
215     term_under_pre, NULL, /* Sx */
216     term_bold_pre, NULL, /* Sy */
217     NULL, NULL, /* Tn */
218     term_xx_pre, NULL, /* Ux */
219     NULL, NULL, /* Xc */
220     NULL, NULL, /* Xo */
221     term_fo_pre, term_fo_post, /* Fo */
222     NULL, NULL, /* Fc */
223     term_quote_pre, term_quote_post, /* Oo */
224     NULL, NULL, /* Oc */
225     term_bk_pre, term_bk_post, /* Bk */
226     NULL, NULL, /* Ek */
227     term_bt_pre, NULL, /* Bt */
228     NULL, NULL, /* Hf */
229     NULL, NULL, /* Fr */
230     term_ud_pre, NULL, /* Ud */
231     NULL, term_lb_post, /* Lb */
232     term_sp_pre, NULL, /* Lp */
233     term_lk_pre, NULL, /* Lk */
234     term_under_pre, NULL, /* Mt */
235     term_quote_pre, term_quote_post, /* Brq */
236     term_quote_pre, term_quote_post, /* Bro */
237     NULL, NULL, /* Brc */
238     NULL, term_post, /* %C */
239     NULL, NULL, /* Es */ /* TODO */
240     NULL, NULL, /* En */ /* TODO */
241     term_xx_pre, NULL, /* Dx */
242     NULL, term_post, /* %Q */
243     term_sp_pre, NULL, /* br */
244     term_sp_pre, NULL, /* sp */
245     term_under_pre, term_post, /* %U */
246     NULL, NULL, /* Ta */
247 };

250 void
251 terminal_mdoc(void *arg, const struct mdoc *mdoc)
252 {
253     const struct mdoc_node *n;
254     const struct mdoc_meta *m;

```

```

255     struct termpp      *p;
257     p = (struct termpp *)arg;
259     if (0 == p->defindent)
260         p->defindent = 5;
262     p->overstep = 0;
263     p->maxrmargin = p->defrmargin;
264     p->tabwidth = term_len(p, 5);
266     if (NULL == p->syntab)
267         p->syntab = mchars_alloc();
269     n = mdoc_node(mdoc);
270     m = mdoc_meta(mdoc);
272     term_begin(p, print_mdoc_head, print_mdoc_foot, m);
274     if (n->child)
275         print_mdoc_nodelist(p, NULL, m, n->child);
277     term_end(p);
278 }

281 static void
282 print_mdoc_nodelist(DECL_ARGS)
283 {
285     print_mdoc_node(p, pair, m, n);
286     if (n->next)
287         print_mdoc_nodelist(p, pair, m, n->next);
288 }

291 /* ARGSUSED */
292 static void
293 print_mdoc_node(DECL_ARGS)
294 {
295     int             chld;
296     const void     *font;
297     struct termpair npair;
298     size_t         offset, rmargin;
300     chld = 1;
301     offset = p->offset;
302     rmargin = p->rmargin;
303     font = term_fontq(p);
305     memset(&npair, 0, sizeof(struct termpair));
306     npair.ppair = pair;
308     /*
309     * Keeps only work until the end of a line.  If a keep was
310     * invoked in a prior line, revert it to PREKEEP.
311     *
312     * Also let SYNPRETTY sections behave as if they were wrapped
313     * in a 'Bk' block.
314     */
316     if (TERMP_KEEP & p->flags || MDOC_SYNPRETTY & n->flags) {
317         if (n->prev && n->prev->line != n->line) {
318             p->flags &= ~TERMP_KEEP;
319             p->flags |= TERMP_PREKEEP;
320         } else if (NULL == n->prev) {

```

```

321         if (n->parent && n->parent->line != n->line) {
322             p->flags &= ~TERMP_KEEP;
323             p->flags |= TERMP_PREKEEP;
324         }
325     }
326 }
328 /*
329 * Since SYNPRETTY sections aren't "turned off" with 'Ek',
330 * we have to intuit whether we should disable formatting.
331 */
333 if ( ! (MDOC_SYNPRETTY & n->flags) &&
334     ((n->prev && MDOC_SYNPRETTY & n->prev->flags) ||
335     (n->parent && MDOC_SYNPRETTY & n->parent->flags)))
336     p->flags &= ~(TERMP_KEEP | TERMP_PREKEEP);
338 /*
339 * After the keep flags have been set up, we may now
340 * produce output.  Note that some pre-handlers do so.
341 */
343 switch (n->type) {
344 case (MDOC_TEXT):
345     if ( ' ' == *n->string && MDOC_LINE & n->flags)
346         term_newln(p);
347     if (MDOC_DELIMC & n->flags)
348         p->flags |= TERMP_NOSPACE;
349     term_word(p, n->string);
350     if (MDOC_DELIMO & n->flags)
351         p->flags |= TERMP_NOSPACE;
352     break;
353 case (MDOC_EQN):
354     term_eqn(p, n->eqn);
355     break;
356 case (MDOC_TBL):
357     term_tbl(p, n->span);
358     break;
359 default:
360     if (termacts[n->tok].pre && ENDBODY_NOT == n->end)
361         chld = (*termacts[n->tok].pre)
362             (p, &npair, m, n);
363     break;
364 }
366 if (chld && n->child)
367     print_mdoc_nodelist(p, &npair, m, n->child);
369 term_fontpopq(p, font);
371 switch (n->type) {
372 case (MDOC_TEXT):
373     break;
374 case (MDOC_TBL):
375     break;
376 case (MDOC_EQN):
377     break;
378 default:
379     if ( ! termacts[n->tok].post || MDOC_ENDED & n->flags)
380         break;
381     (void)(*termacts[n->tok].post)(p, &npair, m, n);
383 /*
384 * Explicit end tokens not only call the post
385 * handler, but also tell the respective block
386 * that it must not call the post handler again.

```

```

387     */
388     if (ENDBODY_NOT != n->end)
389         n->pending->flags |= MDOC_ENDED;

391     /*
392     * End of line terminating an implicit block
393     * while an explicit block is still open.
394     * Continue the explicit block without spacing.
395     */
396     if (ENDBODY_NOSPACE == n->end)
397         p->flags |= TERMP_NOSPACE;
398     break;
399 }

401 if (MDOC_EOS & n->flags)
402     p->flags |= TERMP_SENTENCE;

404 p->offset = offset;
405 p->rmargin = rmargin;
406 }

409 static void
410 print_mdoc_foot(struct term *p, const void *arg)
411 {
412     const struct mdoc_meta *m;

414     m = (const struct mdoc_meta *)arg;

416     term_fontrepl(p, TERMFONT_NONE);

418     /*
419     * Output the footer in new-groff style, that is, three columns
420     * with the middle being the manual date and flanking columns
421     * being the operating system:
422     *
423     * SYSTEM                DATE                SYSTEM
424     */

426     term_vspace(p);

428     p->offset = 0;
429     p->rmargin = (p->maxrmargin -
430                 term_strlen(p, m->date) + term_len(p, 1)) / 2;
431     p->flags |= TERMP_NOSPACE | TERMP_NOBREAK;

433     term_word(p, m->os);
434     term_flushln(p);

436     p->offset = p->rmargin;
437     p->rmargin = p->maxrmargin - term_strlen(p, m->os);
438     p->flags |= TERMP_NOSPACE;

440     term_word(p, m->date);
441     term_flushln(p);

443     p->offset = p->rmargin;
444     p->rmargin = p->maxrmargin;
445     p->flags &= ~TERMP_NOBREAK;
446     p->flags |= TERMP_NOSPACE;

448     term_word(p, m->os);
449     term_flushln(p);

451     p->offset = 0;
452     p->rmargin = p->maxrmargin;

```

```

453     p->flags = 0;
454 }

457 static void
458 print_mdoc_head(struct term *p, const void *arg)
459 {
460     char        buf[BUFSIZ], title[BUFSIZ];
461     size_t      buflen, titlen;
462     const struct mdoc_meta *m;

464     m = (const struct mdoc_meta *)arg;

466     /*
467     * The header is strange. It has three components, which are
468     * really two with the first duplicated. It goes like this:
469     *
470     * IDENTIFIER                TITLE                IDENTIFIER
471     *
472     * The IDENTIFIER is NAME(SECTION), which is the command-name
473     * (if given, or "unknown" if not) followed by the manual page
474     * section. These are given in 'Dt'. The TITLE is a free-form
475     * string depending on the manual volume. If not specified, it
476     * switches on the manual section.
477     */

479     p->offset = 0;
480     p->rmargin = p->maxrmargin;

482     assert(m->vol);
483     strcpy(buf, m->vol, BUFSIZ);
484     buflen = term_strlen(p, buf);

486     if (m->arch) {
487         strcat(buf, " (", BUFSIZ);
488         strcat(buf, m->arch, BUFSIZ);
489         strcat(buf, ")", BUFSIZ);
490     }

492     snprintf(title, BUFSIZ, "%s(%s)", m->title, m->msec);
493     titlen = term_strlen(p, title);

495     p->flags |= TERMP_NOBREAK | TERMP_NOSPACE;
496     p->offset = 0;
497     p->rmargin = 2 * (titlen+1) + buflen < p->maxrmargin ?
498         (p->maxrmargin -
499          term_strlen(p, buf) + term_len(p, 1)) / 2 :
500         p->maxrmargin - buflen;

502     term_word(p, title);
503     term_flushln(p);

505     p->flags |= TERMP_NOSPACE;
506     p->offset = p->rmargin;
507     p->rmargin = p->offset + buflen + titlen < p->maxrmargin ?
508         p->maxrmargin - titlen : p->maxrmargin;

510     term_word(p, buf);
511     term_flushln(p);

513     p->flags &= ~TERMP_NOBREAK;
514     if (p->rmargin + titlen <= p->maxrmargin) {
515         p->flags |= TERMP_NOSPACE;
516         p->offset = p->rmargin;
517         p->rmargin = p->maxrmargin;
518         term_word(p, title);

```



```

519         term_flushln(p);
520     }

522     p->flags &= ~TERMP_NOSPACE;
523     p->offset = 0;
524     p->rmargin = p->maxrmargin;
525 }

528 static size_t
529 a2height(const struct term *p, const char *v)
530 {
531     struct roffsu    su;

534     assert(v);
535     if ( ! a2roffsu(v, &su, SCALE_VS))
536         SCALE_VS_INIT(&su, atoi(v));

538     return(term_vspan(p, &su));
539 }

542 static size_t
543 a2width(const struct term *p, const char *v)
544 {
545     struct roffsu    su;

547     assert(v);
548     if ( ! a2roffsu(v, &su, SCALE_MAX))
549         SCALE_HS_INIT(&su, term_strlen(p, v));

551     return(term_hspan(p, &su));
552 }

555 static size_t
556 a2offs(const struct term *p, const char *v)
557 {
558     struct roffsu    su;

560     if ('\0' == *v)
561         return(0);
562     else if (0 == strcmp(v, "left"))
563         return(0);
564     else if (0 == strcmp(v, "indent"))
565         return(term_len(p, p->defindent + 1));
566     else if (0 == strcmp(v, "indent-two"))
567         return(term_len(p, (p->defindent + 1) * 2));
568     else if ( ! a2roffsu(v, &su, SCALE_MAX))
569         SCALE_HS_INIT(&su, term_strlen(p, v));

571     return(term_hspan(p, &su));
572 }

575 /*
576 * Determine how much space to print out before block elements of 'It'
577 * (and thus 'Bl') and 'Bd'.  And then go ahead and print that space,
578 * too.
579 */
580 static void
581 print_bvspace(struct term *p,
582              const struct mdoc_node *bl,
583              const struct mdoc_node *n)
584 {

```

```

585     const struct mdoc_node *nn;

587     assert(n);

589     term_newln(p);

591     if (MDOC_Bd == bl->tok && bl->norm->Bd.comp)
592         return;
593     if (MDOC_Bl == bl->tok && bl->norm->Bl.comp)
594         return;

596     /* Do not vspace directly after Ss/Sh. */

598     for (nn = n; nn; nn = nn->parent) {
599         if (MDOC_BLOCK != nn->type)
600             continue;
601         if (MDOC_Ss == nn->tok)
602             return;
603         if (MDOC_Sh == nn->tok)
604             return;
605         if (NULL == nn->prev)
606             continue;
607         break;
608     }

610     /* A '-column' does not assert vspace within the list. */

612     if (MDOC_Bl == bl->tok && LIST_column == bl->norm->Bl.type)
613         if (n->prev && MDOC_It == n->prev->tok)
614             return;

616     /* A '-diag' without body does not vspace. */

618     if (MDOC_Bl == bl->tok && LIST_diag == bl->norm->Bl.type)
619         if (n->prev && MDOC_It == n->prev->tok) {
620             assert(n->prev->body);
621             if (NULL == n->prev->body->child)
622                 return;
623         }

625     term_vspace(p);
626 }

629 /* ARGSUSED */
630 static int
631 term_it_pre(DECL_ARGS)
632 {
633     const struct mdoc_node *bl, *nn;
634     char buf[7];
635     int i;
636     size_t width, offset, ncols, dcol;
637     enum mdoc_list type;

639     if (MDOC_BLOCK == n->type) {
640         print_bvspace(p, n->parent->parent, n);
641         return(1);
642     }

644     bl = n->parent->parent->parent;
645     type = bl->norm->Bl.type;

647     /*
648     * First calculate width and offset.  This is pretty easy unless
649     * we're a -column list, in which case all prior columns must
650     * be accounted for.

```

```

651      */
652
653      width = offset = 0;
654
655      if (bl->norm->Bl.offss)
656          offset = a2offs(p, bl->norm->Bl.offss);
657
658      switch (type) {
659      case (LIST_column):
660          if (MDOC_HEAD == n->type)
661              break;
662
663          /*
664           * Imitate groff's column handling:
665           * - For each earlier column, add its width.
666           * - For less than 5 columns, add four more blanks per
667           *   column.
668           * - For exactly 5 columns, add three more blank per
669           *   column.
670           * - For more than 5 columns, add only one column.
671           */
672          ncols = bl->norm->Bl.ncols;
673
674          /* LINTED */
675          dcol = ncols < 5 ? term_len(p, 4) :
676                ncols == 5 ? term_len(p, 3) : term_len(p, 1);
677
678          /*
679           * Calculate the offset by applying all prior MDOC_BODY,
680           * so we stop at the MDOC_HEAD (NULL == nn->prev).
681           */
682          for (i = 0, nn = n->prev;
683              nn->prev && i < (int)ncols;
684              nn = nn->prev, i++)
685              offset += dcol + a2width
686                  (p, bl->norm->Bl.cols[i]);
687
688          /*
689           * When exceeding the declared number of columns, leave
690           * the remaining widths at 0. This will later be
691           * adjusted to the default width of 10, or, for the last
692           * column, stretched to the right margin.
693           */
694          if (i >= (int)ncols)
695              break;
696
697          /*
698           * Use the declared column widths, extended as explained
699           * in the preceding paragraph.
700           */
701          width = a2width(p, bl->norm->Bl.cols[i]) + dcol;
702          break;
703      default:
704          if (NULL == bl->norm->Bl.width)
705              break;
706
707          /*
708           * Note: buffer the width by 2, which is groff's magic
709           * number for buffering single arguments. See the above
710           * handling for column for how this changes.
711           */
712          assert(bl->norm->Bl.width);
713          width = a2width(p, bl->norm->Bl.width) + term_len(p, 2);
714          break;
715      }
716

```

```

718      /*
719       * List-type can override the width in the case of fixed-head
720       * values (bullet, dash/hyphen, enum). Tags need a non-zero
721       * offset.
722       */
723
724      switch (type) {
725      case (LIST_bullet):
726          /* FALLTHROUGH */
727      case (LIST_dash):
728          /* FALLTHROUGH */
729      case (LIST_hyphen):
730          if (width < term_len(p, 4))
731              width = term_len(p, 4);
732          break;
733      case (LIST_enum):
734          if (width < term_len(p, 5))
735              width = term_len(p, 5);
736          break;
737      case (LIST_hang):
738          if (0 == width)
739              width = term_len(p, 8);
740          break;
741      case (LIST_column):
742          /* FALLTHROUGH */
743      case (LIST_tag):
744          if (0 == width)
745              width = term_len(p, 10);
746          break;
747      default:
748          break;
749      }
750
751      /*
752       * Whitespace control. Inset bodies need an initial space,
753       * while diagonal bodies need two.
754       */
755
756      p->flags |= TERMP_NOSPACE;
757
758      switch (type) {
759      case (LIST_diag):
760          if (MDOC_BODY == n->type)
761              term_word(p, "\\ \\ ");
762          break;
763      case (LIST_inset):
764          if (MDOC_BODY == n->type)
765              term_word(p, "\\ ");
766          break;
767      default:
768          break;
769      }
770
771      p->flags |= TERMP_NOSPACE;
772
773      switch (type) {
774      case (LIST_diag):
775          if (MDOC_HEAD == n->type)
776              term_fontpush(p, TERMFONT_BOLD);
777          break;
778      default:
779          break;
780      }
781
782      /*

```

```

783 * Pad and break control. This is the tricky part. These flags
784 * are documented in term_flushln() in term.c. Note that we're
785 * going to unset all of these flags in term_it_post() when we
786 * exit.
787 */
789 switch (type) {
790 case (LIST_bullet):
791     /* FALLTHROUGH */
792 case (LIST_dash):
793     /* FALLTHROUGH */
794 case (LIST_enum):
795     /* FALLTHROUGH */
796 case (LIST_hyphen):
797     if (MDOC_HEAD == n->type)
798         p->flags |= TERMP_NOBREAK;
799     break;
800 case (LIST_hang):
801     if (MDOC_HEAD == n->type)
802         p->flags |= TERMP_NOBREAK;
803     else
804         break;
806     /*
807     * This is ugly. If '-hang' is specified and the body
808     * is a 'B1' or 'Bd', then we want basically to nullify
809     * the "overstep" effect in term_flushln() and treat
810     * this as a '-chang' list instead.
811     */
812     if (n->next->child &&
813         (MDOC_B1 == n->next->child->tok ||
814          MDOC_Bd == n->next->child->tok))
815         p->flags &= ~TERMP_NOBREAK;
816     else
817         p->flags |= TERMP_HANG;
818     break;
819 case (LIST_tag):
820     if (MDOC_HEAD == n->type)
821         p->flags |= TERMP_NOBREAK | TERMP_TWOSPACE;
823     if (MDOC_HEAD != n->type)
824         break;
825     if (NULL == n->next || NULL == n->next->child)
826         p->flags |= TERMP_DANGLE;
827     break;
828 case (LIST_column):
829     if (MDOC_HEAD == n->type)
830         break;
832     if (NULL == n->next)
833         p->flags &= ~TERMP_NOBREAK;
834     else
835         p->flags |= TERMP_NOBREAK;
837     break;
838 case (LIST_diag):
839     if (MDOC_HEAD == n->type)
840         p->flags |= TERMP_NOBREAK;
841     break;
842 default:
843     break;
844 }
846 /*
847 * Margin control. Set-head-width lists have their right
848 * margins shortened. The body for these lists has the offset

```

```

849 * necessarily lengthened. Everybody gets the offset.
850 */
852 p->offset += offset;
854 switch (type) {
855 case (LIST_hang):
856     /*
857     * Same stipulation as above, regarding '-hang'. We
858     * don't want to recalculate rmargin and offsets when
859     * using 'Bd' or 'B1' within '-hang' overstep lists.
860     */
861     if (MDOC_HEAD == n->type && n->next->child &&
862         (MDOC_B1 == n->next->child->tok ||
863          MDOC_Bd == n->next->child->tok))
864         break;
865     /* FALLTHROUGH */
866 case (LIST_bullet):
867     /* FALLTHROUGH */
868 case (LIST_dash):
869     /* FALLTHROUGH */
870 case (LIST_enum):
871     /* FALLTHROUGH */
872 case (LIST_hyphen):
873     /* FALLTHROUGH */
874 case (LIST_tag):
875     assert(width);
876     if (MDOC_HEAD == n->type)
877         p->rmargin = p->offset + width;
878     else
879         p->offset += width;
880     break;
881 case (LIST_column):
882     assert(width);
883     p->rmargin = p->offset + width;
884     /*
885     * XXX - this behaviour is not documented: the
886     * right-most column is filled to the right margin.
887     */
888     if (MDOC_HEAD == n->type)
889         break;
890     if (NULL == n->next && p->rmargin < p->maxrmargin)
891         p->rmargin = p->maxrmargin;
892     break;
893 default:
894     break;
895 }
897 /*
898 * The dash, hyphen, bullet and enum lists all have a special
899 * HEAD character (temporarily bold, in some cases).
900 */
902 if (MDOC_HEAD == n->type)
903     switch (type) {
904     case (LIST_bullet):
905         term_fontpush(p, TERMFONT_BOLD);
906         term_word(p, "\\[bu]");
907         term_fontpop(p);
908         break;
909     case (LIST_dash):
910         /* FALLTHROUGH */
911     case (LIST_hyphen):
912         term_fontpush(p, TERMFONT_BOLD);
913         term_word(p, "\\(hy)");
914         term_fontpop(p);

```

```

915         break;
916     case (LIST_enum):
917         (pair->ppair->ppair->count)++;
918         snprintf(buf, sizeof(buf), "%d.",
919                 pair->ppair->ppair->count);
920         term_word(p, buf);
921         break;
922     default:
923         break;
924 }

926 /*
927  * If we're not going to process our children, indicate so here.
928  */

930 switch (type) {
931 case (LIST_bullet):
932     /* FALLTHROUGH */
933 case (LIST_item):
934     /* FALLTHROUGH */
935 case (LIST_dash):
936     /* FALLTHROUGH */
937 case (LIST_hyphen):
938     /* FALLTHROUGH */
939 case (LIST_enum):
940     if (MDOC_HEAD == n->type)
941         return(0);
942     break;
943 case (LIST_column):
944     if (MDOC_HEAD == n->type)
945         return(0);
946     break;
947 default:
948     break;
949 }

951 return(1);
952 }

955 /* ARGSUSED */
956 static void
957 term_it_post(DECL_ARGS)
958 {
959     enum mdoc_list    type;

961     if (MDOC_BLOCK == n->type)
962         return;

964     type = n->parent->parent->parent->norm->Bl.type;

966     switch (type) {
967 case (LIST_item):
968     /* FALLTHROUGH */
969 case (LIST_diag):
970     /* FALLTHROUGH */
971 case (LIST_inset):
972     if (MDOC_BODY == n->type)
973         term_newln(p);
974     break;
975 case (LIST_column):
976     if (MDOC_BODY == n->type)
977         term_flushln(p);
978     break;
979 default:
980     term_newln(p);

```

```

981         break;
982     }

984     /*
985     * Now that our output is flushed, we can reset our tags.  Since
986     * only 'It' sets these flags, we're free to assume that nobody
987     * has munged them in the meanwhile.
988     */

990     p->flags &= ~TERMP_DANGLE;
991     p->flags &= ~TERMP_NOBREAK;
992     p->flags &= ~TERMP_TWOSPACE;
993     p->flags &= ~TERMP_HANG;
994 }

997 /* ARGSUSED */
998 static int
999 term_nm_pre(DECL_ARGS)
1000 {

1002     if (MDOC_BLOCK == n->type)
1003         return(1);

1005     if (MDOC_BODY == n->type) {
1006         if (NULL == n->child)
1007             return(0);
1008         p->flags |= TERMP_NOSPACE;
1009         p->offset += term_len(p, 1) +
1010             (NULL == n->prev->child ? term_strlen(p, m->name) :
1011              MDOC_TEXT == n->prev->child->type ?
1012               term_strlen(p, n->prev->child->string) :
1013               term_len(p, 5));
1014         return(1);
1015     }

1017     if (NULL == n->child && NULL == m->name)
1018         return(0);

1020     if (MDOC_HEAD == n->type)
1021         synopsis_pre(p, n->parent);

1023     if (MDOC_HEAD == n->type && n->next->child) {
1024         p->flags |= TERMP_NOSPACE | TERMP_NOBREAK;
1025         p->rmargin = p->offset + term_len(p, 1);
1026         if (NULL == n->child) {
1027             p->rmargin += term_strlen(p, m->name);
1028         } else if (MDOC_TEXT == n->child->type) {
1029             p->rmargin += term_strlen(p, n->child->string);
1030             if (n->child->next)
1031                 p->flags |= TERMP_HANG;
1032         } else {
1033             p->rmargin += term_len(p, 5);
1034             p->flags |= TERMP_HANG;
1035         }
1036     }

1038     term_fontpush(p, TERMFONT_BOLD);
1039     if (NULL == n->child)
1040         term_word(p, m->name);
1041     return(1);
1042 }

1045 /* ARGSUSED */
1046 static void

```

```

1047 term_nm_post(DECL_ARGS)
1048 {
1050     if (MDOC_HEAD == n->type && n->next->child) {
1051         term_flushln(p);
1052         p->flags &= ~(TERMP_NOBREAK | TERMP_HANG);
1053     } else if (MDOC_BODY == n->type && n->child)
1054         term_flushln(p);
1055 }
1057
1058 /* ARGSUSED */
1059 static int
1060 term_fl_pre(DECL_ARGS)
1061 {
1063     term_fontpush(p, TERMFONT_BOLD);
1064     term_word(p, "\\-");
1066     if (n->child)
1067         p->flags |= TERMP_NOSPACE;
1068     else if (n->next && n->next->line == n->line)
1069         p->flags |= TERMP_NOSPACE;
1071     return(1);
1072 }
1075 /* ARGSUSED */
1076 static int
1077 term_a_pre(DECL_ARGS)
1078 {
1080     if (n->prev && MDOC_A == n->prev->tok)
1081         if (NULL == n->next || MDOC_A != n->next->tok)
1082             term_word(p, "and");
1084     return(1);
1085 }
1088 /* ARGSUSED */
1089 static int
1090 term_an_pre(DECL_ARGS)
1091 {
1093     if (NULL == n->child)
1094         return(1);
1096     /*
1097     * If not in the AUTHORS section, 'An -split' will cause
1098     * newlines to occur before the author name. If in the AUTHORS
1099     * section, by default, the first 'An' invocation is nosplit,
1100     * then all subsequent ones, regardless of whether interspersed
1101     * with other macros/text, are split. -split, in this case,
1102     * will override the condition of the implied first -nosplit.
1103     */
1104     if (n->sec == SEC_AUTHORS) {
1105         if ( ! (TERMP_ANPREC & p->flags)) {
1106             if (TERMP_SPLIT & p->flags)
1107                 term_newln(p);
1108             return(1);
1109         }
1110     }
1111     if (TERMP_NOSPLIT & p->flags)
1112         return(1);

```

```

1113         term_newln(p);
1114         return(1);
1115     }
1117     if (TERMP_SPLIT & p->flags)
1118         term_newln(p);
1120     return(1);
1121 }
1124 /* ARGSUSED */
1125 static void
1126 term_an_post(DECL_ARGS)
1127 {
1129     if (n->child) {
1130         if (SEC_AUTHORS == n->sec)
1131             p->flags |= TERMP_ANPREC;
1132         return;
1133     }
1135     if (AUTH_split == n->norm->An.auth) {
1136         p->flags &= ~TERMP_NOSPLIT;
1137         p->flags |= TERMP_SPLIT;
1138     } else if (AUTH_nosplit == n->norm->An.auth) {
1139         p->flags &= ~TERMP_SPLIT;
1140         p->flags |= TERMP_NOSPLIT;
1141     }
1143 }
1146 /* ARGSUSED */
1147 static int
1148 term_ns_pre(DECL_ARGS)
1149 {
1151     if ( ! (MDOC_LINE & n->flags))
1152         p->flags |= TERMP_NOSPACE;
1153     return(1);
1154 }
1157 /* ARGSUSED */
1158 static int
1159 term_rs_pre(DECL_ARGS)
1160 {
1162     if (SEC_SEE_ALSO != n->sec)
1163         return(1);
1164     if (MDOC_BLOCK == n->type && n->prev)
1165         term_vspace(p);
1166     return(1);
1167 }
1170 /* ARGSUSED */
1171 static int
1172 term_rv_pre(DECL_ARGS)
1173 {
1174     int             nchild;
1176     term_newln(p);
1177     term_word(p, "The");

```

```

1179     nchild = n->nchild;
1180     for (n = n->child; n; n = n->next) {
1181         term_fontpush(p, TERMFONT_BOLD);
1182         term_word(p, n->string);
1183         term_fontpop(p);
1184
1185         p->flags |= TERMP_NOSPACE;
1186         term_word(p, "(");
1187
1188         if (nchild > 2 && n->next) {
1189             p->flags |= TERMP_NOSPACE;
1190             term_word(p, ",");
1191         }
1192
1193         if (n->next && NULL == n->next->next)
1194             term_word(p, "and");
1195     }
1196
1197     if (nchild > 1)
1198         term_word(p, "functions return");
1199     else
1200         term_word(p, "function returns");
1201
1202     term_word(p, "the value 0 if successful; otherwise the value "
1203              "-1 is returned and the global variable");
1204
1205     term_fontpush(p, TERMFONT_UNDER);
1206     term_word(p, "errno");
1207     term_fontpop(p);
1208
1209     term_word(p, "is set to indicate the error.");
1210     p->flags |= TERMP_SENTENCE;
1211
1212     return(0);
1213 }
1214
1215
1216 /* ARGSUSED */
1217 static int
1218 term_ex_pre(DECL_ARGS)
1219 {
1220     int            nchild;
1221
1222     term_newln(p);
1223     term_word(p, "The");
1224
1225     nchild = n->nchild;
1226     for (n = n->child; n; n = n->next) {
1227         term_fontpush(p, TERMFONT_BOLD);
1228         term_word(p, n->string);
1229         term_fontpop(p);
1230
1231         if (nchild > 2 && n->next) {
1232             p->flags |= TERMP_NOSPACE;
1233             term_word(p, ",");
1234         }
1235
1236         if (n->next && NULL == n->next->next)
1237             term_word(p, "and");
1238     }
1239
1240     if (nchild > 1)
1241         term_word(p, "utilities exit");
1242     else
1243         term_word(p, "utility exits");

```

```

1245         term_word(p, "0 on success, and >0 if an error occurs.");
1246
1247         p->flags |= TERMP_SENTENCE;
1248         return(0);
1249     }
1250
1251
1252 /* ARGSUSED */
1253 static int
1254 term_nd_pre(DECL_ARGS)
1255 {
1256     if (MDOC_BODY != n->type)
1257         return(1);
1258
1259     #if defined(__OpenBSD__) || defined(__linux__)
1260         term_word(p, "\\(en");
1261     #else
1262         term_word(p, "\\(em");
1263     #endif
1264     return(1);
1265 }
1266
1267
1268 /* ARGSUSED */
1269 static int
1270 term_bl_pre(DECL_ARGS)
1271 {
1272     return(MDOC_HEAD != n->type);
1273 }
1274
1275
1276
1277 /* ARGSUSED */
1278 static void
1279 term_bl_post(DECL_ARGS)
1280 {
1281     if (MDOC_BLOCK == n->type)
1282         term_newln(p);
1283 }
1284
1285
1286
1287 /* ARGSUSED */
1288 static int
1289 term_xr_pre(DECL_ARGS)
1290 {
1291     if (NULL == (n = n->child))
1292         return(0);
1293
1294     assert(MDOC_TEXT == n->type);
1295     term_word(p, n->string);
1296
1297     if (NULL == (n = n->next))
1298         return(0);
1299
1300     p->flags |= TERMP_NOSPACE;
1301     term_word(p, "(");
1302     p->flags |= TERMP_NOSPACE;
1303
1304     assert(MDOC_TEXT == n->type);
1305     term_word(p, n->string);
1306
1307     p->flags |= TERMP_NOSPACE;
1308     term_word(p, ")");

```

```

1311     return(0);
1312 }

1314 /*
1315 * This decides how to assert whitespace before any of the SYNOPSIS set
1316 * of macros (which, as in the case of Ft/Fo and Ft/Fn, may contain
1317 * macro combos).
1318 */
1319 static void
1320 synopsis_pre(struct term *p, const struct mdoc_node *n)
1321 {
1322     /*
1323      * Obviously, if we're not in a SYNOPSIS or no prior macros
1324      * exist, do nothing.
1325      */
1326     if (NULL == n->prev || ! (MDOC_SYNPRETTY & n->flags))
1327         return;

1329     /*
1330      * If we're the second in a pair of like elements, emit our
1331      * newline and return. UNLESS we're 'Fo', 'Fn', 'Fn', in which
1332      * case we soldier on.
1333      */
1334     if (n->prev->tok == n->tok &&
1335         MDOC_Ft != n->tok &&
1336         MDOC_Fo != n->tok &&
1337         MDOC_Fn != n->tok) {
1338         term_newln(p);
1339         return;
1340     }

1342     /*
1343      * If we're one of the SYNOPSIS set and non-like pair-wise after
1344      * another (or Fn/Fo, which we've let slip through) then assert
1345      * vertical space, else only newline and move on.
1346      */
1347     switch (n->prev->tok) {
1348     case (MDOC_Fd):
1349         /* FALLTHROUGH */
1350     case (MDOC_Fn):
1351         /* FALLTHROUGH */
1352     case (MDOC_Fo):
1353         /* FALLTHROUGH */
1354     case (MDOC_In):
1355         /* FALLTHROUGH */
1356     case (MDOC_Vt):
1357         term_vspace(p);
1358         break;
1359     case (MDOC_Ft):
1360         if (MDOC_Fn != n->tok && MDOC_Fo != n->tok) {
1361             term_vspace(p);
1362             break;
1363         }
1364         /* FALLTHROUGH */
1365     default:
1366         term_newln(p);
1367         break;
1368     }
1369 }

1372 static int
1373 term_vt_pre(DECL_ARGS)
1374 {
1376     if (MDOC_ELEM == n->type) {

```

```

1377         synopsis_pre(p, n);
1378         return(term_under_pre(p, pair, m, n));
1379     } else if (MDOC_BLOCK == n->type) {
1380         synopsis_pre(p, n);
1381         return(1);
1382     } else if (MDOC_HEAD == n->type)
1383         return(0);

1385     return(term_under_pre(p, pair, m, n));
1386 }

1389 /* ARGSUSED */
1390 static int
1391 term_bold_pre(DECL_ARGS)
1392 {
1394     term_fontpush(p, TERMFONT_BOLD);
1395     return(1);
1396 }

1399 /* ARGSUSED */
1400 static int
1401 term_fd_pre(DECL_ARGS)
1402 {
1404     synopsis_pre(p, n);
1405     return(term_bold_pre(p, pair, m, n));
1406 }

1409 /* ARGSUSED */
1410 static int
1411 term_sh_pre(DECL_ARGS)
1412 {
1414     /* No vspace between consecutive 'Sh' calls. */

1416     switch (n->type) {
1417     case (MDOC_BLOCK):
1418         if (n->prev && MDOC_Sh == n->prev->tok)
1419             if (NULL == n->prev->body->child)
1420                 break;
1421             term_vspace(p);
1422             break;
1423     case (MDOC_HEAD):
1424         term_fontpush(p, TERMFONT_BOLD);
1425         break;
1426     case (MDOC_BODY):
1427         p->offset = term_len(p, p->defindent);
1428         break;
1429     default:
1430         break;
1431     }
1432     return(1);
1433 }

1436 /* ARGSUSED */
1437 static void
1438 term_sh_post(DECL_ARGS)
1439 {
1441     switch (n->type) {
1442     case (MDOC_HEAD):

```

```

1443         term_newln(p);
1444         break;
1445     case (MDOC_BODY):
1446         term_newln(p);
1447         p->offset = 0;
1448         break;
1449     default:
1450         break;
1451 }
1452 }

1455 /* ARGSUSED */
1456 static int
1457 term_bt_pre(DECL_ARGS)
1458 {
1460     term_word(p, "is currently in beta test.");
1461     p->flags |= TERMP_SENTENCE;
1462     return(0);
1463 }

1466 /* ARGSUSED */
1467 static void
1468 term_lb_post(DECL_ARGS)
1469 {
1471     if (SEC_LIBRARY == n->sec && MDOC_LINE & n->flags)
1472         term_newln(p);
1473 }

1476 /* ARGSUSED */
1477 static int
1478 term_ud_pre(DECL_ARGS)
1479 {
1481     term_word(p, "currently under development.");
1482     p->flags |= TERMP_SENTENCE;
1483     return(0);
1484 }

1487 /* ARGSUSED */
1488 static int
1489 term_dl_pre(DECL_ARGS)
1490 {
1492     if (MDOC_BLOCK != n->type)
1493         return(1);
1494     term_newln(p);
1495     p->offset += term_len(p, p->defindent + 1);
1496     return(1);
1497 }

1500 /* ARGSUSED */
1501 static void
1502 term_dl_post(DECL_ARGS)
1503 {
1505     if (MDOC_BLOCK != n->type)
1506         return;
1507     term_newln(p);
1508 }

```

```

1511 /* ARGSUSED */
1512 static int
1513 term_ft_pre(DECL_ARGS)
1514 {
1516     /* NB: MDOC_LINE does not effect this! */
1517     synopsis_pre(p, n);
1518     term_fontpush(p, TERMFONT_UNDER);
1519     return(1);
1520 }

1523 /* ARGSUSED */
1524 static int
1525 term_fn_pre(DECL_ARGS)
1526 {
1527     int         pretty;

1529     pretty = MDOC_SYNPRETTY & n->flags;

1531     synopsis_pre(p, n);

1533     if (NULL == (n = n->child))
1534         return(0);

1536     assert(MDOC_TEXT == n->type);
1537     term_fontpush(p, TERMFONT_BOLD);
1538     term_word(p, n->string);
1539     term_fontpop(p);

1541     p->flags |= TERMP_NOSPACE;
1542     term_word(p, "(");
1543     p->flags |= TERMP_NOSPACE;

1545     for (n = n->next; n; n = n->next) {
1546         assert(MDOC_TEXT == n->type);
1547         term_fontpush(p, TERMFONT_UNDER);
1548         term_word(p, n->string);
1549         term_fontpop(p);

1551         if (n->next) {
1552             p->flags |= TERMP_NOSPACE;
1553             term_word(p, ",");
1554         }
1555     }

1557     p->flags |= TERMP_NOSPACE;
1558     term_word(p, ")");

1560     if (pretty) {
1561         p->flags |= TERMP_NOSPACE;
1562         term_word(p, ";");
1563     }

1565     return(0);
1566 }

1569 /* ARGSUSED */
1570 static int
1571 term_fa_pre(DECL_ARGS)
1572 {
1573     const struct mdoc_node *nn;

```



```

1575     if (n->parent->tok != MDOC_Fo) {
1576         term_fontpush(p, TERMFONT_UNDER);
1577         return(1);
1578     }

1580     for (nn = n->child; nn; nn = nn->next) {
1581         term_fontpush(p, TERMFONT_UNDER);
1582         term_word(p, nn->string);
1583         term_fontpop(p);

1585         if (nn->next) {
1586             p->flags |= TERMP_NOSPACE;
1587             term_word(p, ",");
1588         }
1589     }

1591     if (n->child && n->next && n->next->tok == MDOC_Fa) {
1592         p->flags |= TERMP_NOSPACE;
1593         term_word(p, ",");
1594     }

1596     return(0);
1597 }

1600 /* ARGSUSED */
1601 static int
1602 term_bd_pre(DECL_ARGS)
1603 {
1604     size_t          tabwidth, rm, rmax;
1605     const struct mdoc_node *nn;

1607     if (MDOC_BLOCK == n->type) {
1608         print_bvspace(p, n, n);
1609         return(1);
1610     } else if (MDOC_HEAD == n->type)
1611         return(0);

1613     if (n->norm->Bd.offss)
1614         p->offset += a2offs(p, n->norm->Bd.offss);

1616     /*
1617     * If -ragged or -filled are specified, the block does nothing
1618     * but change the indentation.  If -unfilled or -literal are
1619     * specified, text is printed exactly as entered in the display:
1620     * for macro lines, a newline is appended to the line.  Blank
1621     * lines are allowed.
1622     */
1623
1624     if (DISP_literal != n->norm->Bd.type &&
1625         DISP_unfilled != n->norm->Bd.type)
1626         return(1);

1628     tabwidth = p->tabwidth;
1629     if (DISP_literal == n->norm->Bd.type)
1630         p->tabwidth = term_len(p, 8);

1632     rm = p->rmargin;
1633     rmax = p->maxrmargin;
1634     p->rmargin = p->maxrmargin = TERM_MAXMARGIN;

1636     for (nn = n->child; nn; nn = nn->next) {
1637         print_mdoc_node(p, pair, m, nn);
1638         /*
1639         * If the printed node flushes its own line, then we
1640         * needn't do it here as well.  This is hacky, but the

```

```

1641         * notion of selective eoln whitespace is pretty dumb
1642         * anyway, so don't sweat it.
1643         */
1644         switch (nn->tok) {
1645             case (MDOC_Sm):
1646                 /* FALLTHROUGH */
1647             case (MDOC_br):
1648                 /* FALLTHROUGH */
1649             case (MDOC_sp):
1650                 /* FALLTHROUGH */
1651             case (MDOC_Bl):
1652                 /* FALLTHROUGH */
1653             case (MDOC_Dl):
1654                 /* FALLTHROUGH */
1655             case (MDOC_Dl):
1656                 /* FALLTHROUGH */
1657             case (MDOC_Lp):
1658                 /* FALLTHROUGH */
1659             case (MDOC_Pp):
1660                 continue;
1661             default:
1662                 break;
1663         }
1664         if (nn->next && nn->next->line == nn->line)
1665             continue;
1666         term_flushln(p);
1667         p->flags |= TERMP_NOSPACE;
1668     }

1670     p->tabwidth = tabwidth;
1671     p->rmargin = rm;
1672     p->maxrmargin = rmax;
1673     return(0);
1674 }

1677 /* ARGSUSED */
1678 static void
1679 term_bd_post(DECL_ARGS)
1680 {
1681     size_t          rm, rmax;

1683     if (MDOC_BODY != n->type)
1684         return;

1686     rm = p->rmargin;
1687     rmax = p->maxrmargin;

1689     if (DISP_literal == n->norm->Bd.type ||
1690         DISP_unfilled == n->norm->Bd.type)
1691         p->rmargin = p->maxrmargin = TERM_MAXMARGIN;

1693     p->flags |= TERMP_NOSPACE;
1694     term_newln(p);

1696     p->rmargin = rm;
1697     p->maxrmargin = rmax;
1698 }

1701 /* ARGSUSED */
1702 static int
1703 term_bx_pre(DECL_ARGS)
1704 {

1706     if (NULL != (n = n->child)) {

```

```

1707         term_word(p, n->string);
1708         p->flags |= TERMP_NOSPACE;
1709         term_word(p, "BSD");
1710     } else {
1711         term_word(p, "BSD");
1712         return(0);
1713     }
1714
1715     if (NULL != (n = n->next)) {
1716         p->flags |= TERMP_NOSPACE;
1717         term_word(p, "-");
1718         p->flags |= TERMP_NOSPACE;
1719         term_word(p, n->string);
1720     }
1721
1722     return(0);
1723 }
1724
1725 /* ARGSUSED */
1726 static int
1727 term_xx_pre(DECL_ARGS)
1728 {
1729     const char    *pp;
1730     int           flags;
1731
1732     pp = NULL;
1733     switch (n->tok) {
1734     case (MDOC_Bsx):
1735         pp = "BSD/OS";
1736         break;
1737     case (MDOC_Dx):
1738         pp = "DragonFly";
1739         break;
1740     case (MDOC_Fx):
1741         pp = "FreeBSD";
1742         break;
1743     case (MDOC_Nx):
1744         pp = "NetBSD";
1745         break;
1746     case (MDOC_Ox):
1747         pp = "OpenBSD";
1748         break;
1749     case (MDOC_Ux):
1750         pp = "UNIX";
1751         break;
1752     default:
1753         break;
1754     }
1755
1756     term_word(p, pp);
1757     if (n->child) {
1758         flags = p->flags;
1759         p->flags |= TERMP_KEEP;
1760         term_word(p, n->child->string);
1761         p->flags = flags;
1762     }
1763     return(0);
1764 }
1765
1766 /* ARGSUSED */
1767 static int
1768 term_igndelim_pre(DECL_ARGS)
1769 {

```

```

1773         p->flags |= TERMP_IGNDELIM;
1774         return(1);
1775     }
1776
1777 /* ARGSUSED */
1778 static void
1779 term_pf_post(DECL_ARGS)
1780 {
1781     p->flags |= TERMP_NOSPACE;
1782 }
1783
1784 /* ARGSUSED */
1785 static int
1786 term_ss_pre(DECL_ARGS)
1787 {
1788     switch (n->type) {
1789     case (MDOC_BLOCK):
1790         term_newln(p);
1791         if (n->prev)
1792             term_vspace(p);
1793         break;
1794     case (MDOC_HEAD):
1795         term_fontpush(p, TERMFONT_BOLD);
1796         p->offset = term_len(p, (p->defindent+1)/2);
1797         break;
1798     default:
1799         break;
1800     }
1801     return(1);
1802 }
1803
1804 /* ARGSUSED */
1805 static void
1806 term_ss_post(DECL_ARGS)
1807 {
1808     if (MDOC_HEAD == n->type)
1809         term_newln(p);
1810 }
1811
1812 /* ARGSUSED */
1813 static int
1814 term_cd_pre(DECL_ARGS)
1815 {
1816     synopsis_pre(p, n);
1817     term_fontpush(p, TERMFONT_BOLD);
1818     return(1);
1819 }
1820
1821 /* ARGSUSED */
1822 static int
1823 term_in_pre(DECL_ARGS)
1824 {
1825     synopsis_pre(p, n);
1826     if (MDOC_SYNPRETTY & n->flags && MDOC_LINE & n->flags) {

```

```

1839         term_fontpush(p, TERMFONT_BOLD);
1840         term_word(p, "#include");
1841         term_word(p, "<");
1842     } else {
1843         term_word(p, "<");
1844         term_fontpush(p, TERMFONT_UNDER);
1845     }
1847     p->flags |= TERMP_NOSPACE;
1848     return(1);
1849 }

1852 /* ARGSUSED */
1853 static void
1854 term_in_post(DECL_ARGS)
1855 {
1857     if (MDOC_SYNPRETTY & n->flags)
1858         term_fontpush(p, TERMFONT_BOLD);

1860     p->flags |= TERMP_NOSPACE;
1861     term_word(p, ">");

1863     if (MDOC_SYNPRETTY & n->flags)
1864         term_fontpop(p);
1865 }

1868 /* ARGSUSED */
1869 static int
1870 term_sp_pre(DECL_ARGS)
1871 {
1872     size_t      i, len;

1874     switch (n->tok) {
1875     case (MDOC_sp):
1876         len = n->child ? a2height(p, n->child->string) : 1;
1877         break;
1878     case (MDOC_br):
1879         len = 0;
1880         break;
1881     default:
1882         len = 1;
1883         break;
1884     }

1886     if (0 == len)
1887         term_newln(p);
1888     for (i = 0; i < len; i++)
1889         term_vspace(p);

1891     return(0);
1892 }

1895 /* ARGSUSED */
1896 static int
1897 term_quote_pre(DECL_ARGS)
1898 {

1900     if (MDOC_BODY != n->type && MDOC_ELEM != n->type)
1901         return(1);

1903     switch (n->tok) {
1904     case (MDOC_Ao):

```

```

1905         /* FALLTHROUGH */
1906     case (MDOC_Aq):
1907         term_word(p, "<");
1908         break;
1909     case (MDOC_Bro):
1910         /* FALLTHROUGH */
1911     case (MDOC_Brq):
1912         term_word(p, "{");
1913         break;
1914     case (MDOC_Oo):
1915         /* FALLTHROUGH */
1916     case (MDOC_Op):
1917         /* FALLTHROUGH */
1918     case (MDOC_Bo):
1919         /* FALLTHROUGH */
1920     case (MDOC_Bq):
1921         term_word(p, "[");
1922         break;
1923     case (MDOC_Do):
1924         /* FALLTHROUGH */
1925     case (MDOC_Dq):
1926         term_word(p, "`");
1927         break;
1928     case (MDOC_Eo):
1929         break;
1930     case (MDOC_Po):
1931         /* FALLTHROUGH */
1932     case (MDOC_Pq):
1933         term_word(p, "(");
1934         break;
1935     case (MDOC_T):
1936         /* FALLTHROUGH */
1937     case (MDOC_Qo):
1938         /* FALLTHROUGH */
1939     case (MDOC_Qq):
1940         term_word(p, "\"");
1941         break;
1942     case (MDOC_Ql):
1943         /* FALLTHROUGH */
1944     case (MDOC_So):
1945         /* FALLTHROUGH */
1946     case (MDOC_Sq):
1947         term_word(p, "");
1948         break;
1949     default:
1950         abort();
1951         /* NOTREACHED */
1952     }

1954     p->flags |= TERMP_NOSPACE;
1955     return(1);
1956 }

1959 /* ARGSUSED */
1960 static void
1961 term_quote_post(DECL_ARGS)
1962 {

1964     if (MDOC_BODY != n->type && MDOC_ELEM != n->type)
1965         return;

1967     p->flags |= TERMP_NOSPACE;

1969     switch (n->tok) {
1970     case (MDOC_Ao):

```

```

1971         /* FALLTHROUGH */
1972     case (MDOC_Aq):
1973         term_word(p, ">");
1974         break;
1975     case (MDOC_Bro):
1976         /* FALLTHROUGH */
1977     case (MDOC_Brq):
1978         term_word(p, " ");
1979         break;
1980     case (MDOC_Oo):
1981         /* FALLTHROUGH */
1982     case (MDOC_Op):
1983         /* FALLTHROUGH */
1984     case (MDOC_Bo):
1985         /* FALLTHROUGH */
1986     case (MDOC_Bq):
1987         term_word(p, "]");
1988         break;
1989     case (MDOC_Do):
1990         /* FALLTHROUGH */
1991     case (MDOC_Dq):
1992         term_word(p, "'");
1993         break;
1994     case (MDOC_Eo):
1995         break;
1996     case (MDOC_Po):
1997         /* FALLTHROUGH */
1998     case (MDOC_Pq):
1999         term_word(p, " ");
2000         break;
2001     case (MDOC_T):
2002         /* FALLTHROUGH */
2003     case (MDOC_Qo):
2004         /* FALLTHROUGH */
2005     case (MDOC_Qq):
2006         term_word(p, "\"");
2007         break;
2008     case (MDOC_Ql):
2009         /* FALLTHROUGH */
2010     case (MDOC_So):
2011         /* FALLTHROUGH */
2012     case (MDOC_Sq):
2013         term_word(p, "");
2014         break;
2015     default:
2016         abort();
2017         /* NOTREACHED */
2018     }
2019 }

```

```

2022 /* ARGSUSED */
2023 static int
2024 term_fo_pre(DECL_ARGS)
2025 {

```

```

2027     if (MDOC_BLOCK == n->type) {
2028         synopsis_pre(p, n);
2029         return(1);
2030     } else if (MDOC_BODY == n->type) {
2031         p->flags |= TERMP_NOSPACE;
2032         term_word(p, "(");
2033         p->flags |= TERMP_NOSPACE;
2034         return(1);
2035     }

```

```

2037         if (NULL == n->child)
2038             return(0);
2040         /* XXX: we drop non-initial arguments as per groff. */
2042         assert(n->child->string);
2043         term_fontpush(p, TERMFONT_BOLD);
2044         term_word(p, n->child->string);
2045         return(0);
2046     }

```

```

2049 /* ARGSUSED */
2050 static void
2051 term_fo_post(DECL_ARGS)
2052 {
2054     if (MDOC_BODY != n->type)
2055         return;
2057     p->flags |= TERMP_NOSPACE;
2058     term_word(p, " ");
2060     if (MDOC_SYNPRETTY & n->flags) {
2061         p->flags |= TERMP_NOSPACE;
2062         term_word(p, "");
2063     }
2064 }

```

```

2067 /* ARGSUSED */
2068 static int
2069 term_bf_pre(DECL_ARGS)
2070 {

```

```

2072     if (MDOC_HEAD == n->type)
2073         return(0);
2074     else if (MDOC_BLOCK != n->type)
2075         return(1);
2077     if (FONT_Em == n->norm->Bf.font)
2078         term_fontpush(p, TERMFONT_UNDER);
2079     else if (FONT_Sy == n->norm->Bf.font)
2080         term_fontpush(p, TERMFONT_BOLD);
2081     else
2082         term_fontpush(p, TERMFONT_NONE);
2084     return(1);
2085 }

```

```

2088 /* ARGSUSED */
2089 static int
2090 term_sm_pre(DECL_ARGS)
2091 {

```

```

2093     assert(n->child && MDOC_TEXT == n->child->type);
2094     if (0 == strcmp("on", n->child->string)) {
2095         if (p->col)
2096             p->flags &= ~TERMP_NOSPACE;
2097         p->flags &= ~TERMP_NONOSPACE;
2098     } else
2099         p->flags |= TERMP_NONOSPACE;

```

```

2101     return(0);
2102 }

```

```

2105 /* ARGSUSED */
2106 static int
2107 term_ap_pre(DECL_ARGS)
2108 {
2110     p->flags |= TERMP_NOSPACE;
2111     term_word(p, "");
2112     p->flags |= TERMP_NOSPACE;
2113     return(1);
2114 }

2117 /* ARGSUSED */
2118 static void
2119 term__post(DECL_ARGS)
2120 {
2122     /*
2123      * Handle lists of authors. In general, print each followed by
2124      * a comma. Don't print the comma if there are only two
2125      * authors.
2126      */
2127     if (MDOC_A == n->tok && n->next && MDOC_A == n->next->tok)
2128         if (NULL == n->next->next || MDOC_A != n->next->next->tok)
2129             if (NULL == n->prev || MDOC_A != n->prev->tok)
2130                 return;

2132     /* TODO: %U. */

2134     if (NULL == n->parent || MDOC_Rs != n->parent->tok)
2135         return;

2137     p->flags |= TERMP_NOSPACE;
2138     if (NULL == n->next) {
2139         term_word(p, ".");
2140         p->flags |= TERMP_SENTENCE;
2141     } else
2142         term_word(p, ",");
2143 }

2146 /* ARGSUSED */
2147 static int
2148 term_li_pre(DECL_ARGS)
2149 {
2151     term_fontpush(p, TERMFONT_NONE);
2152     return(1);
2153 }

2156 /* ARGSUSED */
2157 static int
2158 term_lk_pre(DECL_ARGS)
2159 {
2160     const struct mdoc_node *nn, *sv;

2162     term_fontpush(p, TERMFONT_UNDER);

2164     nn = sv = n->child;

2166     if (NULL == nn || NULL == nn->next)
2167         return(1);

```

```

2169         for (nn = nn->next; nn; nn = nn->next)
2170             term_word(p, nn->string);

2172     term_fontpop(p);

2174     p->flags |= TERMP_NOSPACE;
2175     term_word(p, ":");

2177     term_fontpush(p, TERMFONT_BOLD);
2178     term_word(p, sv->string);
2179     term_fontpop(p);

2181     return(0);
2182 }

2185 /* ARGSUSED */
2186 static int
2187 term_bk_pre(DECL_ARGS)
2188 {
2190     switch (n->type) {
2191     case (MDOC_BLOCK):
2192         break;
2193     case (MDOC_HEAD):
2194         return(0);
2195     case (MDOC_BODY):
2196         if (n->parent->args || 0 == n->prev->nchild)
2197             p->flags |= TERMP_PREKEEP;
2198         break;
2199     default:
2200         abort();
2201         /* NOTREACHED */
2202     }

2204     return(1);
2205 }

2208 /* ARGSUSED */
2209 static void
2210 term_bk_post(DECL_ARGS)
2211 {
2213     if (MDOC_BODY == n->type)
2214         p->flags &= ~(TERMP_KEEP | TERMP_PREKEEP);
2215 }

2217 /* ARGSUSED */
2218 static void
2219 term_t_post(DECL_ARGS)
2220 {
2222     /*
2223      * If we're in an 'Rs' and there's a journal present, then quote
2224      * us instead of underlining us (for disambiguation).
2225      */
2226     if (n->parent && MDOC_Rs == n->parent->tok &&
2227         n->parent->norm->Rs.quote_T)
2228         term_quote_post(p, pair, m, n);

2230     term__post(p, pair, m, n);
2231 }

2233 /* ARGSUSED */
2234 static int

```

```
2235 term__t_pre(DECL_ARGS)
2236 {
2238     /*
2239     * If we're in an 'Rs' and there's a journal present, then quote
2240     * us instead of underlining us (for disambiguation).
2241     */
2242     if (n->parent && MDOC_Rs == n->parent->tok &&
2243         n->parent->norm->Rs.quote_T)
2244         return(term_quote_pre(p, pair, m, n));
2246     term_fontpush(p, TERMFONT_UNDER);
2247     return(1);
2248 }
2250 /* ARGSUSED */
2251 static int
2252 term_under_pre(DECL_ARGS)
2253 {
2255     term_fontpush(p, TERMFONT_UNDER);
2256     return(1);
2257 }
```

```

*****
51934 Sat Jul 19 19:24:31 2014
new/usr/src/cmd/mandoc/mdoc_validate.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: mdoc_validate.c,v 1.182 2012/03/23 05:50:25 kristaps Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2010, 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #ifndef OSNAME
23 #include <sys/utsname.h>
24 #endif
25
26 #include <sys/types.h>
27
28 #include <assert.h>
29 #include <ctype.h>
30 #include <limits.h>
31 #include <stdio.h>
32 #include <stdlib.h>
33 #include <string.h>
34 #include <time.h>
35
36 #include "mdoc.h"
37 #include "mandoc.h"
38 #include "libmdoc.h"
39 #include "libmandoc.h"
40
41 /* FIXME: .Bl -diag can't have non-text children in HEAD. */
42
43 #define PRE_ARGS struct mdoc *mdoc, struct mdoc_node *n
44 #define POST_ARGS struct mdoc *mdoc
45
46 #define NUMSIZ 32
47 #define DATESIZE 32
48
49 enum check_ineq {
50     CHECK_LT,
51     CHECK_GT,
52     CHECK_EQ
53 };
54
55 enum check_lvl {
56     CHECK_WARN,

```

```

57     CHECK_ERROR,
58 };
59
60 typedef int (*v_pre)(PRE_ARGS);
61 typedef int (*v_post)(POST_ARGS);
62
63 struct valids {
64     v_pre *pre;
65     v_post *post;
66 };
67
68 static int check_count(struct mdoc *, enum mdoc_type,
69     enum check_lvl, enum check_ineq, int);
70 static int check_parent(PRE_ARGS, enum mdoct, enum mdoc_type);
71 static void check_text(struct mdoc *, int, int, char *);
72 static void check_argv(struct mdoc *,
73     struct mdoc_node *, struct mdoc_argv *);
74 static void check_args(struct mdoc *, struct mdoc_node *);
75 static int concat(char *, const struct mdoc_node *, size_t);
76 static enum mdoc_sec a2sec(const char *);
77 static size_t macro2len(enum mdoct);
78
79 static int ebool(POST_ARGS);
80 static int berr_gel(POST_ARGS);
81 static int bwarn_gel(POST_ARGS);
82 static int ewarn_eq0(POST_ARGS);
83 static int ewarn_eq1(POST_ARGS);
84 static int ewarn_gel(POST_ARGS);
85 static int ewarn_le1(POST_ARGS);
86 static int hwarn_eq0(POST_ARGS);
87 static int hwarn_eq1(POST_ARGS);
88 static int hwarn_gel(POST_ARGS);
89 static int hwarn_le1(POST_ARGS);
90
91 static int post_an(POST_ARGS);
92 static int post_at(POST_ARGS);
93 static int post_bf(POST_ARGS);
94 static int post_bl(POST_ARGS);
95 static int post_bl_block(POST_ARGS);
96 static int post_bl_block_width(POST_ARGS);
97 static int post_bl_block_tag(POST_ARGS);
98 static int post_bl_head(POST_ARGS);
99 static int post_bx(POST_ARGS);
100 static int post_dd(POST_ARGS);
101 static int post_dt(POST_ARGS);
102 static int post_defaults(POST_ARGS);
103 static int post_literal(POST_ARGS);
104 static int post_eoln(POST_ARGS);
105 static int post_it(POST_ARGS);
106 static int post_lb(POST_ARGS);
107 static int post_nm(POST_ARGS);
108 static int post_ns(POST_ARGS);
109 static int post_os(POST_ARGS);
110 static int post_ignpar(POST_ARGS);
111 static int post_prol(POST_ARGS);
112 static int post_root(POST_ARGS);
113 static int post_rs(POST_ARGS);
114 static int post_sh(POST_ARGS);
115 static int post_sh_body(POST_ARGS);
116 static int post_sh_head(POST_ARGS);
117 static int post_st(POST_ARGS);
118 static int post_std(POST_ARGS);
119 static int post_vt(POST_ARGS);
120 static int pre_an(PRE_ARGS);
121 static int pre_bd(PRE_ARGS);
122 static int pre_bl(PRE_ARGS);

```

```

123 static int pre_dd(PRE_ARGS);
124 static int pre_display(PRE_ARGS);
125 static int pre_dt(PRE_ARGS);
126 static int pre_it(PRE_ARGS);
127 static int pre_literal(PRE_ARGS);
128 static int pre_os(PRE_ARGS);
129 static int pre_par(PRE_ARGS);
130 static int pre_sh(PRE_ARGS);
131 static int pre_ss(PRE_ARGS);
132 static int pre_std(PRE_ARGS);

134 static v_post posts_an[] = { post_an, NULL };
135 static v_post posts_at[] = { post_at, post_defaults, NULL };
136 static v_post posts_bd[] = { post_literal, hwarn_eq0, bwarn_gel, NULL };
137 static v_post posts_bf[] = { hwarn_lel, post_bf, NULL };
138 static v_post posts_bk[] = { hwarn_eq0, bwarn_gel, NULL };
139 static v_post posts_bl[] = { bwarn_gel, post_bl, NULL };
140 static v_post posts_bx[] = { post_bx, NULL };
141 static v_post posts_bool[] = { ebool, NULL };
142 static v_post posts_eoln[] = { post_eoln, NULL };
143 static v_post posts_defaults[] = { post_defaults, NULL };
144 static v_post posts_dd[] = { post_dd, post_prol, NULL };
145 static v_post posts_dl[] = { post_literal, bwarn_gel, NULL };
146 static v_post posts_dt[] = { post_dt, post_prol, NULL };
147 static v_post posts_fo[] = { hwarn_eql, bwarn_gel, NULL };
148 static v_post posts_it[] = { post_it, NULL };
149 static v_post posts_lb[] = { post_lb, NULL };
150 static v_post posts_nd[] = { berr_gel, NULL };
151 static v_post posts_nm[] = { post_nm, NULL };
152 static v_post posts_notext[] = { ewarn_eq0, NULL };
153 static v_post posts_ns[] = { post_ns, NULL };
154 static v_post posts_os[] = { post_os, post_prol, NULL };
155 static v_post posts_rs[] = { post_rs, NULL };
156 static v_post posts_sh[] = { post_ignpar, hwarn_gel, post_sh, NULL };
157 static v_post posts_sp[] = { ewarn_lel, NULL };
158 static v_post posts_ss[] = { post_ignpar, hwarn_gel, NULL };
159 static v_post posts_st[] = { post_st, NULL };
160 static v_post posts_std[] = { post_std, NULL };
161 static v_post posts_text[] = { ewarn_gel, NULL };
162 static v_post posts_textl[] = { ewarn_eql, NULL };
163 static v_post posts_vt[] = { post_vt, NULL };
164 static v_post posts_wline[] = { bwarn_gel, NULL };
165 static v_pre pres_an[] = { pre_an, NULL };
166 static v_pre pres_bd[] = { pre_display, pre_bd, pre_literal, pre_par, NULL };
167 static v_pre pres_bl[] = { pre_bl, pre_par, NULL };
168 static v_pre pres_dl[] = { pre_display, NULL };
169 static v_pre pres_dl[] = { pre_literal, pre_display, NULL };
170 static v_pre pres_dd[] = { pre_dd, NULL };
171 static v_pre pres_dt[] = { pre_dt, NULL };
172 static v_pre pres_er[] = { NULL, NULL };
173 static v_pre pres_fd[] = { NULL, NULL };
174 static v_pre pres_it[] = { pre_it, pre_par, NULL };
175 static v_pre pres_os[] = { pre_os, NULL };
176 static v_pre pres_pp[] = { pre_par, NULL };
177 static v_pre pres_sh[] = { pre_sh, NULL };
178 static v_pre pres_ss[] = { pre_ss, NULL };
179 static v_pre pres_std[] = { pre_std, NULL };

181 static const struct valids mdoc_valids[MDOC_MAX] = {
182     { NULL, NULL }, /* Ap */
183     { pres_dd, posts_dd }, /* Dd */
184     { pres_dt, posts_dt }, /* Dt */
185     { pres_os, posts_os }, /* Os */
186     { pres_sh, posts_sh }, /* Sh */
187     { pres_ss, posts_ss }, /* Ss */
188     { pres_pp, posts_notext }, /* Pp */

```

```

189     { pres_dl, posts_wline }, /* Dl */
190     { pres_dl, posts_dl }, /* Dl */
191     { pres_bd, posts_bd }, /* Bd */
192     { NULL, NULL }, /* Ed */
193     { pres_bl, posts_bl }, /* Bl */
194     { NULL, NULL }, /* El */
195     { pres_it, posts_it }, /* It */
196     { NULL, NULL }, /* Ad */
197     { pres_an, posts_an }, /* An */
198     { NULL, posts_defaults }, /* Ar */
199     { NULL, NULL }, /* Cd */
200     { NULL, NULL }, /* Cm */
201     { NULL, NULL }, /* Dv */
202     { pres_er, NULL }, /* Er */
203     { NULL, NULL }, /* Ev */
204     { pres_std, posts_std }, /* Ex */
205     { NULL, NULL }, /* Fa */
206     { pres_fd, posts_text }, /* Fd */
207     { NULL, NULL }, /* Fl */
208     { NULL, NULL }, /* Fn */
209     { NULL, NULL }, /* Ft */
210     { NULL, NULL }, /* Ic */
211     { NULL, posts_textl }, /* In */
212     { NULL, posts_defaults }, /* Li */
213     { NULL, posts_nd }, /* Nd */
214     { NULL, posts_nm }, /* Nm */
215     { NULL, NULL }, /* Op */
216     { NULL, NULL }, /* Ot */
217     { NULL, posts_defaults }, /* Pa */
218     { pres_std, posts_std }, /* Rv */
219     { NULL, posts_st }, /* St */
220     { NULL, NULL }, /* Va */
221     { NULL, posts_vt }, /* Vt */
222     { NULL, posts_text }, /* Xr */
223     { NULL, posts_text }, /* %A */
224     { NULL, posts_text }, /* %B */ /* FIXME: can be used o
225     { NULL, posts_text }, /* %D */
226     { NULL, posts_text }, /* %I */
227     { NULL, posts_text }, /* %J */
228     { NULL, posts_text }, /* %N */
229     { NULL, posts_text }, /* %O */
230     { NULL, posts_text }, /* %P */
231     { NULL, posts_text }, /* %R */
232     { NULL, posts_text }, /* %T */ /* FIXME: can be used o
233     { NULL, posts_text }, /* %V */
234     { NULL, NULL }, /* Ac */
235     { NULL, NULL }, /* Ao */
236     { NULL, NULL }, /* Aq */
237     { NULL, posts_at }, /* At */
238     { NULL, NULL }, /* Bc */
239     { NULL, posts_bf }, /* Bf */
240     { NULL, NULL }, /* Bo */
241     { NULL, NULL }, /* Bq */
242     { NULL, NULL }, /* Bsx */
243     { NULL, posts_bx }, /* Bx */
244     { NULL, posts_bool }, /* Db */
245     { NULL, NULL }, /* Dc */
246     { NULL, NULL }, /* Do */
247     { NULL, NULL }, /* Dq */
248     { NULL, NULL }, /* Ec */
249     { NULL, NULL }, /* Ef */
250     { NULL, NULL }, /* Em */
251     { NULL, NULL }, /* Eo */
252     { NULL, NULL }, /* Fx */
253     { NULL, NULL }, /* Ms */
254     { NULL, posts_notext }, /* No */

```



```

255     { NULL, posts_ns },          /* Ns */
256     { NULL, NULL },           /* Nx */
257     { NULL, NULL },           /* Ox */
258     { NULL, NULL },           /* Pc */
259     { NULL, posts_text1 },     /* Pf */
260     { NULL, NULL },           /* Po */
261     { NULL, NULL },           /* Pq */
262     { NULL, NULL },           /* Qc */
263     { NULL, NULL },           /* Ql */
264     { NULL, NULL },           /* Qo */
265     { NULL, NULL },           /* Qq */
266     { NULL, NULL },           /* Re */
267     { NULL, posts_rs },        /* Rs */
268     { NULL, NULL },           /* Sc */
269     { NULL, NULL },           /* So */
270     { NULL, NULL },           /* Sq */
271     { NULL, posts_bool },      /* Sm */
272     { NULL, NULL },           /* Sx */
273     { NULL, NULL },           /* Sy */
274     { NULL, NULL },           /* Tn */
275     { NULL, NULL },           /* Ux */
276     { NULL, NULL },           /* Xc */
277     { NULL, NULL },           /* Xo */
278     { NULL, posts_fo },        /* Fo */
279     { NULL, NULL },           /* Fc */
280     { NULL, NULL },           /* Oo */
281     { NULL, NULL },           /* Oc */
282     { NULL, posts_bk },        /* Bk */
283     { NULL, NULL },           /* Ek */
284     { NULL, posts_eoln },      /* Bt */
285     { NULL, NULL },           /* Hf */
286     { NULL, NULL },           /* Fr */
287     { NULL, posts_eoln },      /* Ud */
288     { NULL, posts_lb },        /* Lb */
289     { NULL, posts_notext },    /* Ip */
290     { NULL, NULL },           /* Lk */
291     { NULL, posts_defaults },  /* Mt */
292     { NULL, NULL },           /* Brq */
293     { NULL, NULL },           /* Bro */
294     { NULL, NULL },           /* Brc */
295     { NULL, posts_text },      /* %C */
296     { NULL, NULL },           /* Es */
297     { NULL, NULL },           /* En */
298     { NULL, NULL },           /* Dx */
299     { NULL, posts_text },      /* %Q */
300     { NULL, posts_notext },    /* br */
301     { pres_pp, posts_sp },      /* sp */
302     { NULL, posts_text1 },     /* %U */
303     { NULL, NULL },           /* Ta */
304 };

306 #define RSORD_MAX 14 /* Number of 'Rs' blocks. */

308 static const enum mdoct rsord[RSORD_MAX] = {
309     MDOC_A,
310     MDOC_T,
311     MDOC_B,
312     MDOC_I,
313     MDOC_J,
314     MDOC_R,
315     MDOC_N,
316     MDOC_V,
317     MDOC_P,
318     MDOC_Q,
319     MDOC_D,
320     MDOC_O,

```

```

321     MDOC_C,
322     MDOC_U
323 };

325 static const char * const secnames[SEC_MAX] = {
326     NULL,
327     "NAME",
328     "LIBRARY",
329     "SYNOPSIS",
330     "DESCRIPTION",
331     "IMPLEMENTATION NOTES",
332     "RETURN VALUES",
333     "ENVIRONMENT",
334     "FILES",
335     "EXIT STATUS",
336     "EXAMPLES",
337     "DIAGNOSTICS",
338     "COMPATIBILITY",
339     "ERRORS",
340     "SEE ALSO",
341     "STANDARDS",
342     "HISTORY",
343     "AUTHORS",
344     "CAVEATS",
345     "BUGS",
346     "SECURITY CONSIDERATIONS",
347     NULL
348 };

350 int
351 mdoc_valid_pre(struct mdoc *mdoc, struct mdoc_node *n)
352 {
353     v_pre      *p;
354     int         line, pos;
355     char        *tp;

357     switch (n->type) {
358     case (MDOC_TEXT):
359         tp = n->string;
360         line = n->line;
361         pos = n->pos;
362         check_text(mdoc, line, pos, tp);
363         /* FALLTHROUGH */
364     case (MDOC_TBL):
365         /* FALLTHROUGH */
366     case (MDOC_EQN):
367         /* FALLTHROUGH */
368     case (MDOC_ROOT):
369         return(1);
370     default:
371         break;
372     }

374     check_args(mdoc, n);

376     if (NULL == mdoc_valids[n->tok].pre)
377         return(1);
378     for (p = mdoc_valids[n->tok].pre; *p; p++)
379         if ( ! (*p)(mdoc, n) )
380             return(0);
381     return(1);
382 }

385 int
386 mdoc_valid_post(struct mdoc *mdoc)

```

```

387 {
388     v_post      *p;

390     if (MDOC_VALID & mdoc->last->flags)
391         return(1);
392     mdoc->last->flags |= MDOC_VALID;

394     switch (mdoc->last->type) {
395     case (MDOC_TEXT):
396         /* FALLTHROUGH */
397     case (MDOC_EQN):
398         /* FALLTHROUGH */
399     case (MDOC_TBL):
400         return(1);
401     case (MDOC_ROOT):
402         return(post_root(mdoc));
403     default:
404         break;
405     }

407     if (NULL == mdoc_valids[mdoc->last->tok].post)
408         return(1);
409     for (p = mdoc_valids[mdoc->last->tok].post; *p; p++)
410         if (!(*p)(mdoc))
411             return(0);

413     return(1);
414 }

416 static int
417 check_count(struct mdoc *m, enum mdoc_type type,
418             enum check_lvl lvl, enum check_ineq ineq, int val)
419 {
420     const char    *p;
421     enum mandocerr t;

423     if (m->last->type != type)
424         return(1);
425
426     switch (ineq) {
427     case (CHECK_LT):
428         p = "less than ";
429         if (m->last->nchild < val)
430             return(1);
431         break;
432     case (CHECK_GT):
433         p = "more than ";
434         if (m->last->nchild > val)
435             return(1);
436         break;
437     case (CHECK_EQ):
438         p = "";
439         if (val == m->last->nchild)
440             return(1);
441         break;
442     default:
443         abort();
444         /* NOTREACHED */
445     }

447     t = lvl == CHECK_WARN ? MANDOCERR_ARGCWARN : MANDOCERR_ARGCOUNT;
448     mandoc_vmsg(t, m->parse, m->last->line, m->last->pos,
449               "want %s%d children (have %d)",
450               p, val, m->last->nchild);
451     return(1);
452 }

```

```

454 static int
455 berr_gel(POST_ARGS)
456 {
458     return(check_count(mdoc, MDOC_BODY, CHECK_ERROR, CHECK_GT, 0));
459 }

461 static int
462 bwarn_gel(POST_ARGS)
463 {
464     return(check_count(mdoc, MDOC_BODY, CHECK_WARN, CHECK_GT, 0));
465 }

467 static int
468 ewarn_eq0(POST_ARGS)
469 {
470     return(check_count(mdoc, MDOC_ELEM, CHECK_WARN, CHECK_EQ, 0));
471 }

473 static int
474 ewarn_eq1(POST_ARGS)
475 {
476     return(check_count(mdoc, MDOC_ELEM, CHECK_WARN, CHECK_EQ, 1));
477 }

479 static int
480 ewarn_gel(POST_ARGS)
481 {
482     return(check_count(mdoc, MDOC_ELEM, CHECK_WARN, CHECK_GT, 0));
483 }

485 static int
486 ewarn_le1(POST_ARGS)
487 {
488     return(check_count(mdoc, MDOC_ELEM, CHECK_WARN, CHECK_LT, 2));
489 }

491 static int
492 hwarn_eq0(POST_ARGS)
493 {
494     return(check_count(mdoc, MDOC_HEAD, CHECK_WARN, CHECK_EQ, 0));
495 }

497 static int
498 hwarn_eq1(POST_ARGS)
499 {
500     return(check_count(mdoc, MDOC_HEAD, CHECK_WARN, CHECK_EQ, 1));
501 }

503 static int
504 hwarn_gel(POST_ARGS)
505 {
506     return(check_count(mdoc, MDOC_HEAD, CHECK_WARN, CHECK_GT, 0));
507 }

509 static int
510 hwarn_le1(POST_ARGS)
511 {
512     return(check_count(mdoc, MDOC_HEAD, CHECK_WARN, CHECK_LT, 2));
513 }

515 static void
516 check_args(struct mdoc *m, struct mdoc_node *n)
517 {
518     int            i;

```

```

520     if (NULL == n->args)
521         return;

523     assert(n->args->argc);
524     for (i = 0; i < (int)n->args->argc; i++)
525         check_argv(m, n, &n->args->argv[i]);
526 }

528 static void
529 check_argv(struct mdoc *m, struct mdoc_node *n, struct mdoc_argv *v)
530 {
531     int            i;

533     for (i = 0; i < (int)v->sz; i++)
534         check_text(m, v->line, v->pos, v->value[i]);

536     /* FIXME: move to post_std(). */

538     if (MDOC_Std == v->arg)
539         if ( ! (v->sz || m->meta.name))
540             mdoc_nmsg(m, n, MANDOCERR_NONAME);
541 }

543 static void
544 check_text(struct mdoc *m, int ln, int pos, char *p)
545 {
546     char            *cp;

548     if (MDOC_LITERAL & m->flags)
549         return;

551     for (cp = p; NULL != (p = strchr(p, '\t')); p++)
552         mdoc_pmsg(m, ln, pos + (int)(p - cp), MANDOCERR_BADTAB);
553 }

555 static int
556 check_parent(PRE_ARGS, enum mdoct tok, enum mdoc_type t)
557 {
559     assert(n->parent);
560     if ((MDOC_ROOT == t || tok == n->parent->tok) &&
561         (t == n->parent->type))
562         return(1);

564     mandoc_vmsg(MANDOCERR_SYNTCHILD, mdoc->parse, n->line,
565                n->pos, "want parent %s", MDOC_ROOT == t ?
566                "<root>" : mdoc_macronames[tok]);
567     return(0);
568 }

571 static int
572 pre_display(PRE_ARGS)
573 {
574     struct mdoc_node *node;

576     if (MDOC_BLOCK != n->type)
577         return(1);

579     for (node = mdoc->last->parent; node; node = node->parent)
580         if (MDOC_BLOCK == node->type)
581             if (MDOC_Bd == node->tok)
582                 break;

584     if (node)

```

```

585         mdoc_nmsg(mdoc, n, MANDOCERR_NESTEDDISP);

587     return(1);
588 }

591 static int
592 pre_bl(PRE_ARGS)
593 {
594     int            i, comp, dup;
595     const char     *offs, *width;
596     enum mdoc_list lt;
597     struct mdoc_node *np;

599     if (MDOC_BLOCK != n->type) {
600         if (ENDBODY_NOT != n->end) {
601             assert(n->pending);
602             np = n->pending->parent;
603         } else
604             np = n->parent;

606         assert(np);
607         assert(MDOC_BLOCK == np->type);
608         assert(MDOC_Bl == np->tok);
609         return(1);
610     }

612     /*
613      * First figure out which kind of list to use: bind ourselves to
614      * the first mentioned list type and warn about any remaining
615      * ones. If we find no list type, we default to LIST_item.
616      */

618     /* LINTED */
619     for (i = 0; n->args && i < (int)n->args->argc; i++) {
620         lt = LIST_NONE;
621         dup = comp = 0;
622         width = offs = NULL;
623         switch (n->args->argv[i].arg) {
624             /* Set list types. */
625             case (MDOC_Bullet):
626                 lt = LIST_bullet;
627                 break;
628             case (MDOC_Dash):
629                 lt = LIST_dash;
630                 break;
631             case (MDOC_Enum):
632                 lt = LIST_enum;
633                 break;
634             case (MDOC_Hyphen):
635                 lt = LIST_hyphen;
636                 break;
637             case (MDOC_Item):
638                 lt = LIST_item;
639                 break;
640             case (MDOC_Tag):
641                 lt = LIST_tag;
642                 break;
643             case (MDOC_Diag):
644                 lt = LIST_diag;
645                 break;
646             case (MDOC_Hang):
647                 lt = LIST_hang;
648                 break;
649             case (MDOC_Ohang):
650                 lt = LIST_ohang;

```

```

651         break;
652     case (MDOC_Inset):
653         lt = LIST_inset;
654         break;
655     case (MDOC_Column):
656         lt = LIST_column;
657         break;
658     /* Set list arguments. */
659     case (MDOC_Compact):
660         dup = n->norm->Bl.comp;
661         comp = 1;
662         break;
663     case (MDOC_Width):
664         /* NB: this can be empty! */
665         if (n->args->argv[i].sz) {
666             width = n->args->argv[i].value[0];
667             dup = (NULL != n->norm->Bl.width);
668             break;
669         }
670         mdoc_nmsg(mdoc, n, MANDOCERR_IGNARGV);
671         break;
672     case (MDOC_Offset):
673         /* NB: this can be empty! */
674         if (n->args->argv[i].sz) {
675             offs = n->args->argv[i].value[0];
676             dup = (NULL != n->norm->Bl.offs);
677             break;
678         }
679         mdoc_nmsg(mdoc, n, MANDOCERR_IGNARGV);
680         break;
681     default:
682         continue;
683 }
684
685 /* Check: duplicate auxiliary arguments. */
686
687 if (dup)
688     mdoc_nmsg(mdoc, n, MANDOCERR_ARGVREP);
689
690 if (comp && ! dup)
691     n->norm->Bl.comp = comp;
692 if (offs && ! dup)
693     n->norm->Bl.offs = offs;
694 if (width && ! dup)
695     n->norm->Bl.width = width;
696
697 /* Check: multiple list types. */
698
699 if (LIST_NONE != lt && n->norm->Bl.type != LIST_NONE)
700     mdoc_nmsg(mdoc, n, MANDOCERR_LISTREP);
701
702 /* Assign list type. */
703
704 if (LIST_NONE != lt && n->norm->Bl.type == LIST_NONE) {
705     n->norm->Bl.type = lt;
706     /* Set column information, too. */
707     if (LIST_column == lt) {
708         n->norm->Bl.ncols =
709             n->args->argv[i].sz;
710         n->norm->Bl.cols = (void *)
711             n->args->argv[i].value;
712     }
713 }
714
715 /* The list type should come first. */

```

```

717         if (n->norm->Bl.type == LIST_NONE)
718             if (n->norm->Bl.width ||
719                 n->norm->Bl.offs ||
720                 n->norm->Bl.comp)
721                 mdoc_nmsg(mdoc, n, MANDOCERR_LISTFIRST);
722
723         continue;
724     }
725
726     /* Allow lists to default to LIST_item. */
727
728     if (LIST_NONE == n->norm->Bl.type) {
729         mdoc_nmsg(mdoc, n, MANDOCERR_LISTTYPE);
730         n->norm->Bl.type = LIST_item;
731     }
732
733     /*
734     * Validate the width field. Some list types don't need width
735     * types and should be warned about them. Others should have it
736     * and must also be warned.
737     */
738
739     switch (n->norm->Bl.type) {
740     case (LIST_tag):
741         if (n->norm->Bl.width)
742             break;
743         mdoc_nmsg(mdoc, n, MANDOCERR_NOWIDTHARG);
744         break;
745     case (LIST_column):
746         /* FALLTHROUGH */
747     case (LIST_diag):
748         /* FALLTHROUGH */
749     case (LIST_ohang):
750         /* FALLTHROUGH */
751     case (LIST_inset):
752         /* FALLTHROUGH */
753     case (LIST_item):
754         if (n->norm->Bl.width)
755             mdoc_nmsg(mdoc, n, MANDOCERR_IGNARGV);
756         break;
757     default:
758         break;
759     }
760
761     return(1);
762 }
763
764
765 static int
766 pre_bd(PRE_ARGS)
767 {
768     int            i, dup, comp;
769     enum mdoc_disp dt;
770     const char    *offs;
771     struct mdoc_node *np;
772
773     if (MDOC_BLOCK != n->type) {
774         if (ENDBODY_NOT != n->end) {
775             assert(n->pending);
776             np = n->pending->parent;
777         } else
778             np = n->parent;
779
780         assert(np);
781         assert(MDOC_BLOCK == np->type);
782         assert(MDOC_Bd == np->tok);

```

```

783     return(1);
784 }

786 /* LINTED */
787 for (i = 0; n->args && i < (int)n->args->argc; i++) {
788     dt = DISP_NONE;
789     dup = comp = 0;
790     offs = NULL;

792     switch (n->args->argv[i].arg) {
793     case (MDOC_Centred):
794         dt = DISP_centred;
795         break;
796     case (MDOC_Ragged):
797         dt = DISP_ragged;
798         break;
799     case (MDOC_Unfilled):
800         dt = DISP_unfilled;
801         break;
802     case (MDOC_Filled):
803         dt = DISP_filled;
804         break;
805     case (MDOC_Literal):
806         dt = DISP_literal;
807         break;
808     case (MDOC_File):
809         mdoc_nmsg(mdoc, n, MANDOCERR_BADDISP);
810         return(0);
811     case (MDOC_Offset):
812         /* NB: this can be empty! */
813         if (n->args->argv[i].sz) {
814             offs = n->args->argv[i].value[0];
815             dup = (NULL != n->norm->Bd.offs);
816             break;
817         }
818         mdoc_nmsg(mdoc, n, MANDOCERR_IGNARGV);
819         break;
820     case (MDOC_Compact):
821         comp = 1;
822         dup = n->norm->Bd.comp;
823         break;
824     default:
825         abort();
826         /* NOTREACHED */
827     }

829     /* Check whether we have duplicates. */

831     if (dup)
832         mdoc_nmsg(mdoc, n, MANDOCERR_ARGVREP);

834     /* Make our auxiliary assignments. */

836     if (offs && ! dup)
837         n->norm->Bd.offs = offs;
838     if (comp && ! dup)
839         n->norm->Bd.comp = comp;

841     /* Check whether a type has already been assigned. */

843     if (DISP_NONE != dt && n->norm->Bd.type != DISP_NONE)
844         mdoc_nmsg(mdoc, n, MANDOCERR_DISPREP);

846     /* Make our type assignment. */

848     if (DISP_NONE != dt && n->norm->Bd.type == DISP_NONE)

```

```

849         n->norm->Bd.type = dt;
850     }

852     if (DISP_NONE == n->norm->Bd.type) {
853         mdoc_nmsg(mdoc, n, MANDOCERR_DISPTYPE);
854         n->norm->Bd.type = DISP_ragged;
855     }

857     return(1);
858 }

861 static int
862 pre_ss(PRE_ARGS)
863 {

865     if (MDOC_BLOCK != n->type)
866         return(1);
867     return(check_parent(mdoc, n, MDOC_Sh, MDOC_BODY));
868 }

871 static int
872 pre_sh(PRE_ARGS)
873 {

875     if (MDOC_BLOCK != n->type)
876         return(1);

878     roff_regunset(mdoc->roff, REG_nS);
879     return(check_parent(mdoc, n, MDOC_MAX, MDOC_ROOT));
880 }

883 static int
884 pre_it(PRE_ARGS)
885 {

887     if (MDOC_BLOCK != n->type)
888         return(1);

890     return(check_parent(mdoc, n, MDOC_Bl, MDOC_BODY));
891 }

894 static int
895 pre_an(PRE_ARGS)
896 {
897     int         i;

899     if (NULL == n->args)
900         return(1);
901

902     for (i = 1; i < (int)n->args->argc; i++)
903         mdoc_pmsg(mdoc, n->args->argv[i].line,
904                 n->args->argv[i].pos, MANDOCERR_IGNARGV);

906     if (MDOC_Split == n->args->argv[0].arg)
907         n->norm->An.auth = AUTH_split;
908     else if (MDOC_Nosplit == n->args->argv[0].arg)
909         n->norm->An.auth = AUTH_nosplit;
910     else
911         abort();

913     return(1);
914 }

```

```

916 static int
917 pre_std(PRE_ARGS)
918 {
919     if (n->args && 1 == n->args->argc)
920         if (MDOC_Std == n->args->argv[0].arg)
921             return(1);
922
923     mdoc_nmsg(mdoc, n, MANDOCERR_NOARGV);
924     return(1);
925 }
926
927 static int
928 pre_dt(PRE_ARGS)
929 {
930     if (NULL == mdoc->meta.date || mdoc->meta.os)
931         mdoc_nmsg(mdoc, n, MANDOCERR_PROLOGOOO);
932
933     if (mdoc->meta.title)
934         mdoc_nmsg(mdoc, n, MANDOCERR_PROLOGREP);
935
936     return(1);
937 }
938
939 static int
940 pre_os(PRE_ARGS)
941 {
942     if (NULL == mdoc->meta.title || NULL == mdoc->meta.date)
943         mdoc_nmsg(mdoc, n, MANDOCERR_PROLOGOOO);
944
945     if (mdoc->meta.os)
946         mdoc_nmsg(mdoc, n, MANDOCERR_PROLOGREP);
947
948     return(1);
949 }
950
951 static int
952 pre_dd(PRE_ARGS)
953 {
954     if (mdoc->meta.title || mdoc->meta.os)
955         mdoc_nmsg(mdoc, n, MANDOCERR_PROLOGOOO);
956
957     if (mdoc->meta.date)
958         mdoc_nmsg(mdoc, n, MANDOCERR_PROLOGREP);
959
960     return(1);
961 }
962
963 static int
964 post_bf(POST_ARGS)
965 {
966     struct mdoc_node *np;
967     enum mdocargt arg;
968
969     /*
970      * Unlike other data pointers, these are "housed" by the HEAD
971      * element, which contains the goods.
972      */
973
974     if (MDOC_HEAD != mdoc->last->type) {
975         if (ENDBODY_NOT != mdoc->last->end) {

```

```

981         assert(mdoc->last->pending);
982         np = mdoc->last->pending->parent->head;
983     } else if (MDOC_BLOCK != mdoc->last->type) {
984         np = mdoc->last->parent->head;
985     } else
986         np = mdoc->last->head;
987
988     assert(np);
989     assert(MDOC_HEAD == np->type);
990     assert(MDOC_Bf == np->tok);
991     return(1);
992 }
993
994 np = mdoc->last;
995 assert(MDOC_BLOCK == np->parent->type);
996 assert(MDOC_Bf == np->parent->tok);
997
998 /*
999  * Cannot have both argument and parameter.
1000  * If neither is specified, let it through with a warning.
1001  */
1002
1003 if (np->parent->args && np->child) {
1004     mdoc_nmsg(mdoc, np, MANDOCERR_SYNTARGVCOUNT);
1005     return(0);
1006 } else if (NULL == np->parent->args && NULL == np->child) {
1007     mdoc_nmsg(mdoc, np, MANDOCERR_FONTTYPE);
1008     return(1);
1009 }
1010
1011 /* Extract argument into data. */
1012
1013 if (np->parent->args) {
1014     arg = np->parent->args->argv[0].arg;
1015     if (MDOC_Emphasis == arg)
1016         np->norm->Bf.font = FONT_Em;
1017     else if (MDOC_Literal == arg)
1018         np->norm->Bf.font = FONT_Li;
1019     else if (MDOC_Symbolic == arg)
1020         np->norm->Bf.font = FONT_Sy;
1021     else
1022         abort();
1023     return(1);
1024 }
1025
1026 /* Extract parameter into data. */
1027
1028 if (0 == strcmp(np->child->string, "Em"))
1029     np->norm->Bf.font = FONT_Em;
1030 else if (0 == strcmp(np->child->string, "Li"))
1031     np->norm->Bf.font = FONT_Li;
1032 else if (0 == strcmp(np->child->string, "Sy"))
1033     np->norm->Bf.font = FONT_Sy;
1034 else
1035     mdoc_nmsg(mdoc, np, MANDOCERR_FONTTYPE);
1036
1037     return(1);
1038 }
1039
1040 static int
1041 post_lb(POST_ARGS)
1042 {
1043     const char *p;
1044     char *buf;
1045     size_t sz;

```

```

1047     check_count(mdoc, MDOC_ELEM, CHECK_WARN, CHECK_EQ, 1);
1049     assert(mdoc->last->child);
1050     assert(MDOC_TEXT == mdoc->last->child->type);
1052     p = mdoc_a2lib(mdoc->last->child->string);
1054     /* If lookup ok, replace with table value. */
1056     if (p) {
1057         free(mdoc->last->child->string);
1058         mdoc->last->child->string = mandoc_strdup(p);
1059         return(1);
1060     }
1062     /* If not, use "library `xxxx'". */
1064     sz = strlen(mdoc->last->child->string) +
1065         2 + strlen("\\(lqlibrary\\(rq");
1066     buf = mandoc_malloc(sz);
1067     snprintf(buf, sz, "library \\(lq%s\\(rq",
1068             mdoc->last->child->string);
1069     free(mdoc->last->child->string);
1070     mdoc->last->child->string = buf;
1071     return(1);
1072 }

1074 static int
1075 post_eoln(POST_ARGS)
1076 {
1078     if (mdoc->last->child)
1079         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_ARGSLOST);
1080     return(1);
1081 }

1084 static int
1085 post_vt(POST_ARGS)
1086 {
1087     const struct mdoc_node *n;
1089     /*
1090     * The Vt macro comes in both ELEM and BLOCK form, both of which
1091     * have different syntaxes (yet more context-sensitive
1092     * behaviour). ELEM types must have a child, which is already
1093     * guaranteed by the in_line parsing routine; BLOCK types,
1094     * specifically the BODY, should only have TEXT children.
1095     */
1097     if (MDOC_BODY != mdoc->last->type)
1098         return(1);
1099     for (n = mdoc->last->child; n; n = n->next)
1100         if (MDOC_TEXT != n->type)
1101             mdoc_nmsg(mdoc, n, MANDOCERR_CHILD);
1104     return(1);
1105 }

1108 static int
1109 post_nm(POST_ARGS)
1110 {
1111     char        buf[BUFSIZ];
1112     int         c;

```

```

1114     /* If no child specified, make sure we have the meta name. */
1116     if (NULL == mdoc->last->child && NULL == mdoc->meta.name) {
1117         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_NONAME);
1118         return(1);
1119     } else if (mdoc->meta.name)
1120         return(1);
1122     /* If no meta name, set it from the child. */
1124     buf[0] = '\0';
1125     if (-1 == (c = concat(buf, mdoc->last->child, BUFSIZ))) {
1126         mdoc_nmsg(mdoc, mdoc->last->child, MANDOCERR_MEM);
1127         return(0);
1128     }
1130     assert(c);
1131     mdoc->meta.name = mandoc_strdup(buf);
1132     return(1);
1133 }

1135 static int
1136 post_literal(POST_ARGS)
1137 {
1138     /*
1139     * The 'Dl' (note "el" not "one") and 'Bd' macros unset the
1140     * MDOC_LITERAL flag as they leave. Note that 'Bd' only sets
1141     * this in literal mode, but it doesn't hurt to just switch it
1142     * off in general since displays can't be nested.
1143     */
1144     if (MDOC_BODY == mdoc->last->type)
1145         mdoc->flags &= ~MDOC_LITERAL;
1149     return(1);
1150 }

1152 static int
1153 post_defaults(POST_ARGS)
1154 {
1155     struct mdoc_node *nn;
1157     /*
1158     * The 'Ar' defaults to "file ..." if no value is provided as an
1159     * argument; the 'Mt' and 'Pa' macros use "~"; the 'Li' just
1160     * gets an empty string.
1161     */
1163     if (mdoc->last->child)
1164         return(1);
1165     nn = mdoc->last;
1166     mdoc->next = MDOC_NEXT_CHILD;
1169     switch (nn->tok) {
1170     case (MDOC_Ar):
1171         if ( ! mdoc_word_alloc(mdoc, nn->line, nn->pos, "file"))
1172             return(0);
1173         if ( ! mdoc_word_alloc(mdoc, nn->line, nn->pos, "..."))
1174             return(0);
1175         break;
1176     case (MDOC_At):
1177         if ( ! mdoc_word_alloc(mdoc, nn->line, nn->pos, "AT&T"))
1178             return(0);

```

```

1179         if ( ! mdoc_word_alloc(mdoc, nn->line, nn->pos, "UNIX"))
1180             return(0);
1181         break;
1182     case (MDOC_Li):
1183         if ( ! mdoc_word_alloc(mdoc, nn->line, nn->pos, ""))
1184             return(0);
1185         break;
1186     case (MDOC_Pa):
1187         /* FALLTHROUGH */
1188     case (MDOC_Mt):
1189         if ( ! mdoc_word_alloc(mdoc, nn->line, nn->pos, "~"))
1190             return(0);
1191         break;
1192     default:
1193         abort();
1194         /* NOTREACHED */
1195     }
1197     mdoc->last = nn;
1198     return(1);
1199 }

```

```

1201 static int
1202 post_at(POST_ARGS)
1203 {
1204     const char    *p, *q;
1205     char          *buf;
1206     size_t        sz;
1208     /*
1209     * If we have a child, look it up in the standard keys.  If a
1210     * key exist, use that instead of the child; if it doesn't,
1211     * prefix "AT&T UNIX " to the existing data.
1212     */
1213     if (NULL == mdoc->last->child)
1214         return(1);

```

```

1217     assert(MDOC_TEXT == mdoc->last->child->type);
1218     p = mdoc_a2att(mdoc->last->child->string);

```

```

1220     if (p) {
1221         free(mdoc->last->child->string);
1222         mdoc->last->child->string = mandoc_strdup(p);
1223     } else {
1224         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_BADATT);
1225         p = "AT&T UNIX ";
1226         q = mdoc->last->child->string;
1227         sz = strlen(p) + strlen(q) + 1;
1228         buf = mandoc_malloc(sz);
1229         strlcpy(buf, p, sz);
1230         strlcat(buf, q, sz);
1231         free(mdoc->last->child->string);
1232         mdoc->last->child->string = buf;
1233     }

```

```

1235     return(1);
1236 }

```

```

1238 static int
1239 post_an(POST_ARGS)
1240 {
1241     struct mdoc_node *np;
1243     np = mdoc->last;
1244     if (AUTH_NONE == np->norm->An.auth) {

```

```

1245         if (0 == np->child)
1246             check_count(mdoc, MDOC_ELEM, CHECK_WARN, CHECK_GT, 0);
1247     } else if (np->child)
1248         check_count(mdoc, MDOC_ELEM, CHECK_WARN, CHECK_EQ, 0);
1250     return(1);
1251 }

```

```

1254 static int
1255 post_it(POST_ARGS)
1256 {
1257     int            i, cols;
1258     enum mdoc_list lt;
1259     struct mdoc_node *n, *c;
1260     enum mandocerr er;
1262     if (MDOC_BLOCK != mdoc->last->type)
1263         return(1);
1265     n = mdoc->last->parent->parent;
1266     lt = n->norm->Bl.type;
1268     if (LIST_NONE == lt) {
1269         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_LISTTYPE);
1270         return(1);
1271     }

```

```

1273     switch (lt) {
1274     case (LIST_tag):
1275         if (mdoc->last->head->child)
1276             break;
1277         /* FIXME: give this a dummy value. */
1278         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_NOARGS);
1279         break;
1280     case (LIST_hang):
1281         /* FALLTHROUGH */
1282     case (LIST_ohang):
1283         /* FALLTHROUGH */
1284     case (LIST_inset):
1285         /* FALLTHROUGH */
1286     case (LIST_diag):
1287         if (NULL == mdoc->last->head->child)
1288             mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_NOARGS);
1289         break;
1290     case (LIST_bullet):
1291         /* FALLTHROUGH */
1292     case (LIST_dash):
1293         /* FALLTHROUGH */
1294     case (LIST_enum):
1295         /* FALLTHROUGH */
1296     case (LIST_hyphen):
1297         if (NULL == mdoc->last->body->child)
1298             mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_NOBODY);
1299         /* FALLTHROUGH */
1300     case (LIST_item):
1301         if (mdoc->last->head->child)
1302             mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_ARGSLOST);
1303         break;
1304     case (LIST_column):
1305         cols = (int)n->norm->Bl.ncols;
1307         assert(NULL == mdoc->last->head->child);
1309         if (NULL == mdoc->last->body->child)
1310             mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_NOBODY);

```



```

1312     for (i = 0, c = mdoc->last->child; c; c = c->next)
1313         if (MDOC_BODY == c->type)
1314             i++;
1315
1316     if (i < cols)
1317         er = MANDOCERR_ARGCOUNT;
1318     else if (i == cols || i == cols + 1)
1319         break;
1320     else
1321         er = MANDOCERR_SYNTARGCOUNT;
1322
1323     mandoc_vmsg(er, mdoc->parse, mdoc->last->line,
1324               mdoc->last->pos,
1325               "columns == %d (have %d)", cols, i);
1326     return(MANDOCERR_ARGCOUNT == er);
1327 default:
1328     break;
1329 }
1330
1331 return(1);
1332 }
1333
1334 static int
1335 post_bl_block(POST_ARGS)
1336 {
1337     struct mdoc_node *n;
1338
1339     /*
1340     * These are fairly complicated, so we've broken them into two
1341     * functions. post_bl_block_tag() is called when a -tag is
1342     * specified, but no -width (it must be guessed). The second
1343     * when a -width is specified (macro indicators must be
1344     * rewritten into real lengths).
1345     */
1346
1347     n = mdoc->last;
1348
1349     if (LIST_tag == n->norm->Bl.type &&
1350         NULL == n->norm->Bl.width) {
1351         if (! post_bl_block_tag(mdoc))
1352             return(0);
1353     } else if (NULL != n->norm->Bl.width) {
1354         if (! post_bl_block_width(mdoc))
1355             return(0);
1356     } else
1357         return(1);
1358
1359     assert(n->norm->Bl.width);
1360     return(1);
1361 }
1362
1363 static int
1364 post_bl_block_width(POST_ARGS)
1365 {
1366     size_t      width;
1367     int         i;
1368     enum mdoc_t tok;
1369     struct mdoc_node *n;
1370     char        buf[NUMSIZ];
1371
1372     n = mdoc->last;
1373
1374     /*
1375     * Calculate the real width of a list from the -width string,
1376     * which may contain a macro (with a known default width), a

```

```

1377     * literal string, or a scaling width.
1378     *
1379     * If the value to -width is a macro, then we re-write it to be
1380     * the macro's width as set in share/tmac/mdoc/doc-common.
1381     */
1382
1383     if (0 == strcmp(n->norm->Bl.width, "Ds"))
1384         width = 6;
1385     else if (MDOC_MAX == (tok = mdoc_hash_find(n->norm->Bl.width)))
1386         return(1);
1387     else if (0 == (width = macro2len(tok))) {
1388         mdoc_nmsg(mdoc, n, MANDOCERR_BADWIDTH);
1389         return(1);
1390     }
1391
1392     /* The value already exists: free and reallocate it. */
1393
1394     assert(n->args);
1395
1396     for (i = 0; i < (int)n->args->argc; i++)
1397         if (MDOC_Width == n->args->argv[i].arg)
1398             break;
1399
1400     assert(i < (int)n->args->argc);
1401
1402     snprintf(buf, NUMSIZ, "%un", (unsigned int)width);
1403     free(n->args->argv[i].value[0]);
1404     n->args->argv[i].value[0] = mandoc_strdup(buf);
1405
1406     /* Set our width! */
1407     n->norm->Bl.width = n->args->argv[i].value[0];
1408     return(1);
1409 }
1410
1411 static int
1412 post_bl_block_tag(POST_ARGS)
1413 {
1414     struct mdoc_node *n, *nn;
1415     size_t          sz, ssize;
1416     int             i;
1417     char            buf[NUMSIZ];
1418
1419     /*
1420     * Calculate the -width for a 'Bl -tag' list if it hasn't been
1421     * provided. Uses the first head macro. NOTE AGAIN: this is
1422     * ONLY if the -width argument has NOT been provided. See
1423     * post_bl_block_width() for converting the -width string.
1424     */
1425
1426     sz = 10;
1427     n = mdoc->last;
1428
1429     for (nn = n->body->child; nn; nn = nn->next) {
1430         if (MDOC_It != nn->tok)
1431             continue;
1432
1433         assert(MDOC_BLOCK == nn->type);
1434         nn = nn->head->child;
1435
1436         if (nn == NULL)
1437             break;
1438
1439         if (MDOC_TEXT == nn->type) {
1440             sz = strlen(nn->string) + 1;
1441             break;
1442         }

```

```

1444         if (0 != (ssz = macro2len(nn->tok)))
1445             sz = ssz;
1447     break;
1448 }
1450 /* Defaults to ten ens. */
1452 snprintf(buf, NUMSIZ, "%un", (unsigned int)sz);
1454 /*
1455  * We have to dynamically add this to the macro's argument list.
1456  * We're guaranteed that a MDOC_Width doesn't already exist.
1457  */
1459 assert(n->args);
1460 i = (int)(n->args->argc)++;
1462 n->args->argv = mdoc_realloc(n->args->argv,
1463     n->args->argc * sizeof(struct mdoc_argv));
1465 n->args->argv[i].arg = MDOC_Width;
1466 n->args->argv[i].line = n->line;
1467 n->args->argv[i].pos = n->pos;
1468 n->args->argv[i].sz = 1;
1469 n->args->argv[i].value = mdoc_malloc(sizeof(char *));
1470 n->args->argv[i].value[0] = mdoc_strdup(buf);
1472 /* Set our width! */
1473 n->norm->Bl.width = n->args->argv[i].value[0];
1474 return(1);
1475 }
1478 static int
1479 post_bl_head(POST_ARGS)
1480 {
1481     struct mdoc_node *np, *nn, *nnp;
1482     int i, j;
1484     if (LIST_column != mdoc->last->norm->Bl.type)
1485         /* FIXME: this should be ERROR class... */
1486         return(hwarn_eq0(mdoc));
1488     /*
1489      * Convert old-style lists, where the column width specifiers
1490      * trail as macro parameters, to the new-style ("normal-form")
1491      * lists where they're argument values following -column.
1492      */
1494     /* First, disallow both types and allow normal-form. */
1496     /*
1497      * TODO: technically, we can accept both and just merge the two
1498      * lists, but I'll leave that for another day.
1499      */
1501     if (mdoc->last->norm->Bl.ncols && mdoc->last->nchild) {
1502         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_COLUMNS);
1503         return(0);
1504     } else if (NULL == mdoc->last->child)
1505         return(1);
1507     np = mdoc->last->parent;
1508     assert(np->args);

```

```

1510     for (j = 0; j < (int)np->args->argc; j++)
1511         if (MDOC_Column == np->args->argv[j].arg)
1512             break;
1514     assert(j < (int)np->args->argc);
1515     assert(0 == np->args->argv[j].sz);
1517     /*
1518      * Accommodate for new-style groff column syntax. Shuffle the
1519      * child nodes, all of which must be TEXT, as arguments for the
1520      * column field. Then, delete the head children.
1521      */
1523     np->args->argv[j].sz = (size_t)mdoc->last->nchild;
1524     np->args->argv[j].value = mdoc_malloc
1525         ((size_t)mdoc->last->nchild * sizeof(char *));
1527     mdoc->last->norm->Bl.ncols = np->args->argv[j].sz;
1528     mdoc->last->norm->Bl.cols = (void *)np->args->argv[j].value;
1530     for (i = 0, nn = mdoc->last->child; nn; i++) {
1531         np->args->argv[j].value[i] = nn->string;
1532         nn->string = NULL;
1533         nnp = nn;
1534         nn = nn->next;
1535         mdoc_node_delete(NULL, nnp);
1536     }
1538     mdoc->last->nchild = 0;
1539     mdoc->last->child = NULL;
1541     return(1);
1542 }
1544 static int
1545 post_bl(POST_ARGS)
1546 {
1547     struct mdoc_node *n;
1549     if (MDOC_HEAD == mdoc->last->type)
1550         return(post_bl_head(mdoc));
1551     if (MDOC_BLOCK == mdoc->last->type)
1552         return(post_bl_block(mdoc));
1553     if (MDOC_BODY != mdoc->last->type)
1554         return(1);
1556     for (n = mdoc->last->child; n; n = n->next) {
1557         switch (n->tok) {
1558             case (MDOC_Lp):
1559                 /* FALLTHROUGH */
1560             case (MDOC_Pp):
1561                 mdoc_nmsg(mdoc, n, MANDOCERR_CHILD);
1562                 /* FALLTHROUGH */
1563             case (MDOC_It):
1564                 /* FALLTHROUGH */
1565             case (MDOC_Sm):
1566                 continue;
1567             default:
1568                 break;
1569         }
1571         mdoc_nmsg(mdoc, n, MANDOCERR_SYNTCHILD);
1572         return(0);
1573     }

```

```

1575     return(1);
1576 }

1578 static int
1579 ebool(struct mdoc *mdoc)
1580 {
1582     if (NULL == mdoc->last->child) {
1583         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_MACROEMPTY);
1584         mdoc_node_delete(mdoc, mdoc->last);
1585         return(1);
1586     }
1587     check_count(mdoc, MDOC_ELEM, CHECK_WARN, CHECK_EQ, 1);

1589     assert(MDOC_TEXT == mdoc->last->child->type);

1591     if (0 == strcmp(mdoc->last->child->string, "on"))
1592         return(1);
1593     if (0 == strcmp(mdoc->last->child->string, "off"))
1594         return(1);

1596     mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_BADBOOL);
1597     return(1);
1598 }

1600 static int
1601 post_root(POST_ARGS)
1602 {
1603     int            erc;
1604     struct mdoc_node *n;

1606     erc = 0;

1608     /* Check that we have a finished prologue. */

1610     if ( ! (MDOC_PBODY & mdoc->flags) ) {
1611         erc++;
1612         mdoc_nmsg(mdoc, mdoc->first, MANDOCERR_NODOCPROLOG);
1613     }

1615     n = mdoc->first;
1616     assert(n);
1617
1618     /* Check that we begin with a proper 'Sh'. */

1620     if (NULL == n->child) {
1621         erc++;
1622         mdoc_nmsg(mdoc, n, MANDOCERR_NODOCBODY);
1623     } else if (MDOC_BLOCK != n->child->type ||
1624                MDOC_Sh != n->child->tok) {
1625         erc++;
1626         /* Can this be lifted? See rxdebug.1 for example. */
1627         mdoc_nmsg(mdoc, n, MANDOCERR_NODOCBODY);
1628     }

1630     return(erc ? 0 : 1);
1631 }

1633 static int
1634 post_st(POST_ARGS)
1635 {
1636     struct mdoc_node    *ch;
1637     const char          *p;

1639     if (NULL == (ch = mdoc->last->child)) {
1640         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_MACROEMPTY);

```

```

1641         mdoc_node_delete(mdoc, mdoc->last);
1642         return(1);
1643     }

1645     assert(MDOC_TEXT == ch->type);

1647     if (NULL == (p = mdoc_a2st(ch->string))) {
1648         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_BADSTANDARD);
1649         mdoc_node_delete(mdoc, mdoc->last);
1650     } else {
1651         free(ch->string);
1652         ch->string = mandoc_strdup(p);
1653     }

1655     return(1);
1656 }

1658 static int
1659 post_rs(POST_ARGS)
1660 {
1661     struct mdoc_node *nn, *next, *prev;
1662     int            i, j;

1664     switch (mdoc->last->type) {
1665     case (MDOC_HEAD):
1666         check_count(mdoc, MDOC_HEAD, CHECK_WARN, CHECK_EQ, 0);
1667         return(1);
1668     case (MDOC_BODY):
1669         if (mdoc->last->child)
1670             break;
1671         check_count(mdoc, MDOC_BODY, CHECK_WARN, CHECK_GT, 0);
1672         return(1);
1673     default:
1674         return(1);
1675     }

1677     /*
1678      * Make sure only certain types of nodes are allowed within the
1679      * the 'Rs' body. Delete offending nodes and raise a warning.
1680      * Do this before re-ordering for the sake of clarity.
1681      */

1683     next = NULL;
1684     for (nn = mdoc->last->child; nn; nn = next) {
1685         for (i = 0; i < RSORD_MAX; i++)
1686             if (nn->tok == rsord[i])
1687                 break;

1689         if (i < RSORD_MAX) {
1690             if (MDOC__J == rsord[i] || MDOC__B == rsord[i])
1691                 mdoc->last->norm->Rs.quote_T++;
1692             next = nn->next;
1693             continue;
1694         }

1696         next = nn->next;
1697         mdoc_nmsg(mdoc, nn, MANDOCERR_CHILD);
1698         mdoc_node_delete(mdoc, nn);
1699     }

1701     /*
1702      * Nothing to sort if only invalid nodes were found
1703      * inside the 'Rs' body.
1704      */

1706     if (NULL == mdoc->last->child)

```

```

1707         return(1);
1709     /*
1710     * The full 'Rs' block needs special handling to order the
1711     * sub-elements according to 'rsord'. Pick through each element
1712     * and correctly order it. This is a insertion sort.
1713     */
1715     next = NULL;
1716     for (nn = mdoc->last->child->next; nn; nn = next) {
1717         /* Determine order of 'nn'. */
1718         for (i = 0; i < RSORD_MAX; i++)
1719             if (rsord[i] == nn->tok)
1720                 break;
1722         /*
1723         * Remove 'nn' from the chain. This somewhat
1724         * repeats mdoc_node_unlink(), but since we're
1725         * just re-ordering, there's no need for the
1726         * full unlink process.
1727         */
1728         if (NULL != (next = nn->next))
1729             next->prev = nn->prev;
1730
1732         if (NULL != (prev = nn->prev))
1733             prev->next = nn->next;
1735         nn->prev = nn->next = NULL;
1737         /*
1738         * Scan back until we reach a node that's
1739         * ordered before 'nn'.
1740         */
1742         for ( ; prev ; prev = prev->prev) {
1743             /* Determine order of 'prev'. */
1744             for (j = 0; j < RSORD_MAX; j++)
1745                 if (rsord[j] == prev->tok)
1746                     break;
1748             if (j <= i)
1749                 break;
1750         }
1752         /*
1753         * Set 'nn' back into its correct place in front
1754         * of the 'prev' node.
1755         */
1757         nn->prev = prev;
1759         if (prev) {
1760             if (prev->next)
1761                 prev->next->prev = nn;
1762             nn->next = prev->next;
1763             prev->next = nn;
1764         } else {
1765             mdoc->last->child->prev = nn;
1766             nn->next = mdoc->last->child;
1767             mdoc->last->child = nn;
1768         }
1769     }
1771     return(1);
1772 }

```

```

1774 static int
1775 post_ns(POST_ARGS)
1776 {
1778     if (MDOC_LINE & mdoc->last->flags)
1779         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_IGNNS);
1780     return(1);
1781 }
1783 static int
1784 post_sh(POST_ARGS)
1785 {
1787     if (MDOC_HEAD == mdoc->last->type)
1788         return(post_sh_head(mdoc));
1789     if (MDOC_BODY == mdoc->last->type)
1790         return(post_sh_body(mdoc));
1792     return(1);
1793 }
1795 static int
1796 post_sh_body(POST_ARGS)
1797 {
1798     struct mdoc_node *n;
1800     if (SEC_NAME != mdoc->lastsec)
1801         return(1);
1803     /*
1804     * Warn if the NAME section doesn't contain the 'Nm' and 'Nd'
1805     * macros (can have multiple 'Nm' and one 'Nd'). Note that the
1806     * children of the BODY declaration can also be "text".
1807     */
1809     if (NULL == (n = mdoc->last->child)) {
1810         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_BADNAMESEC);
1811         return(1);
1812     }
1814     for ( ; n && n->next; n = n->next) {
1815         if (MDOC_ELEM == n->type && MDOC_Nm == n->tok)
1816             continue;
1817         if (MDOC_TEXT == n->type)
1818             continue;
1819         mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_BADNAMESEC);
1820     }
1822     assert(n);
1823     if (MDOC_BLOCK == n->type && MDOC_Nd == n->tok)
1824         return(1);
1826     mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_BADNAMESEC);
1827     return(1);
1828 }
1830 static int
1831 post_sh_head(POST_ARGS)
1832 {
1833     char          buf[BUFSIZ];
1834     struct mdoc_node *n;
1835     enum mdoc_sec  sec;
1836     int           c;
1838     /*

```

```

1839  * Process a new section. Sections are either "named" or
1840  * "custom". Custom sections are user-defined, while named ones
1841  * follow a conventional order and may only appear in certain
1842  * manual sections.
1843  */
1844
1845  sec = SEC_CUSTOM;
1846  buf[0] = '\0';
1847  if (-1 == (c = concat(buf, mdoc->last->child, BUFSIZ))) {
1848      mdoc_nmsg(mdoc, mdoc->last->child, MANDOCERR_MEM);
1849      return(0);
1850  } else if (1 == c)
1851      sec = a2sec(buf);
1852
1853  /* The NAME should be first. */
1854
1855  if (SEC_NAME != sec && SEC_NONE == mdoc->lastnamed)
1856      mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_NAMESECFIRST);
1857
1858  /* The SYNOPSIS gets special attention in other areas. */
1859
1860  if (SEC_SYNOPSIS == sec)
1861      mdoc->flags |= MDOC_SYNOPSIS;
1862  else
1863      mdoc->flags &= ~MDOC_SYNOPSIS;
1864
1865  /* Mark our last section. */
1866
1867  mdoc->lastsec = sec;
1868
1869  /*
1870  * Set the section attribute for the current HEAD, for its
1871  * parent BLOCK, and for the HEAD children; the latter can
1872  * only be TEXT nodes, so no recursion is needed.
1873  * For other blocks and elements, including .Sh BODY, this is
1874  * done when allocating the node data structures, but for .Sh
1875  * BLOCK and HEAD, the section is still unknown at that time.
1876  */
1877
1878  mdoc->last->parent->sec = sec;
1879  mdoc->last->sec = sec;
1880  for (n = mdoc->last->child; n; n = n->next)
1881      n->sec = sec;
1882
1883  /* We don't care about custom sections after this. */
1884
1885  if (SEC_CUSTOM == sec)
1886      return(1);
1887
1888  /*
1889  * Check whether our non-custom section is being repeated or is
1890  * out of order.
1891  */
1892
1893  if (sec == mdoc->lastnamed)
1894      mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_SECREP);
1895
1896  if (sec < mdoc->lastnamed)
1897      mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_SECOOO);
1898
1899  /* Mark the last named section. */
1900
1901  mdoc->lastnamed = sec;
1902
1903  /* Check particular section/manual conventions. */

```

```

1905      assert(mdoc->meta.msec);
1906
1907      switch (sec) {
1908      case (SEC_RETURN_VALUES):
1909          /* FALLTHROUGH */
1910      case (SEC_ERRORS):
1911          /* FALLTHROUGH */
1912      case (SEC_LIBRARY):
1913          if (*mdoc->meta.msec == '2')
1914              break;
1915          if (*mdoc->meta.msec == '3')
1916              break;
1917          if (*mdoc->meta.msec == '9')
1918              break;
1919          mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_SECMSEC);
1920          break;
1921      default:
1922          break;
1923      }
1924
1925      return(1);
1926  }
1927
1928  static int
1929  post_ignpar(POST_ARGS)
1930  {
1931      struct mdoc_node *np;
1932
1933      if (MDOC_BODY != mdoc->last->type)
1934          return(1);
1935
1936      if (NULL != (np = mdoc->last->child))
1937          if (MDOC_Pp == np->tok || MDOC_Lp == np->tok) {
1938              mdoc_nmsg(mdoc, np, MANDOCERR_IGNPAR);
1939              mdoc_node_delete(mdoc, np);
1940          }
1941
1942      if (NULL != (np = mdoc->last->last))
1943          if (MDOC_Pp == np->tok || MDOC_Lp == np->tok) {
1944              mdoc_nmsg(mdoc, np, MANDOCERR_IGNPAR);
1945              mdoc_node_delete(mdoc, np);
1946          }
1947
1948      return(1);
1949  }
1950
1951  static int
1952  pre_par(PRE_ARGS)
1953  {
1954
1955      if (NULL == mdoc->last)
1956          return(1);
1957      if (MDOC_ELEM != n->type && MDOC_BLOCK != n->type)
1958          return(1);
1959
1960      /*
1961      * Don't allow prior 'Lp' or 'Pp' prior to a paragraph-type
1962      * block: 'Lp', 'Pp', or non-compact 'Bd' or 'Bl'.
1963      */
1964
1965      if (MDOC_Pp != mdoc->last->tok && MDOC_Lp != mdoc->last->tok)
1966          return(1);
1967      if (MDOC_Bl == n->tok && n->norm->Bl.comp)
1968          return(1);
1969      if (MDOC_Bd == n->tok && n->norm->Bd.comp)
1970          return(1);

```

```

1971     if (MDOC_It == n->tok && n->parent->norm->Bl.comp)
1972         return(1);

1974     mdoc_nmsg(mdoc, mdoc->last, MANDOCERR_IGNPAR);
1975     mdoc_node_delete(mdoc, mdoc->last);
1976     return(1);
1977 }

1979 static int
1980 pre_literal(PRE_ARGS)
1981 {
1983     if (MDOC_BODY != n->type)
1984         return(1);

1986     /*
1987      * The 'Dl' (note "el" not "one") and 'Bd -literal' and 'Bd
1988      * -unfilled' macros set MDOC_LITERAL on entrance to the body.
1989      */

1991     switch (n->tok) {
1992     case (MDOC_Dl):
1993         mdoc->flags |= MDOC_LITERAL;
1994         break;
1995     case (MDOC_Bd):
1996         if (DISP_literal == n->norm->Bd.type)
1997             mdoc->flags |= MDOC_LITERAL;
1998         if (DISP_unfilled == n->norm->Bd.type)
1999             mdoc->flags |= MDOC_LITERAL;
2000         break;
2001     default:
2002         abort();
2003         /* NOTREACHED */
2004     }
2005
2006     return(1);
2007 }

2009 static int
2010 post_dd(POST_ARGS)
2011 {
2012     char        buf[DATESIZE];
2013     struct mdoc_node *n;
2014     int         c;

2016     if (mdoc->meta.date)
2017         free(mdoc->meta.date);

2019     n = mdoc->last;
2020     if (NULL == n->child || '\0' == n->child->string[0]) {
2021         mdoc->meta.date = mandoc_normdate
2022             (mdoc->parse, NULL, n->line, n->pos);
2023         return(1);
2024     }

2026     buf[0] = '\0';
2027     if (-1 == (c = concat(buf, n->child, DATESIZE))) {
2028         mdoc_nmsg(mdoc, n->child, MANDOCERR_MEM);
2029         return(0);
2030     }

2032     assert(c);
2033     mdoc->meta.date = mandoc_normdate
2034         (mdoc->parse, buf, n->line, n->pos);

2036     return(1);

```

```

2037 }

2039 static int
2040 post_dt(POST_ARGS)
2041 {
2042     struct mdoc_node *nn, *n;
2043     const char      *cp;
2044     char            *p;

2046     n = mdoc->last;

2048     if (mdoc->meta.title)
2049         free(mdoc->meta.title);
2050     if (mdoc->meta.vol)
2051         free(mdoc->meta.vol);
2052     if (mdoc->meta.arch)
2053         free(mdoc->meta.arch);

2055     mdoc->meta.title = mdoc->meta.vol = mdoc->meta.arch = NULL;

2057     /* First make all characters uppercase. */

2059     if (NULL != (nn = n->child))
2060         for (p = nn->string; *p; p++) {
2061             if (toupper((unsigned char)*p) == *p)
2062                 continue;

2064             /*
2065              * FIXME: don't be lazy: have this make all
2066              * characters be uppercase and just warn once.
2067              */
2068             mdoc_nmsg(mdoc, nn, MANDOCERR_UPPERCASE);
2069             break;
2070         }

2072     /* Handles: '.Dt'
2073      * --> title = unknown, volume = local, msec = 0, arch = NULL
2074      */

2076     if (NULL == (nn = n->child)) {
2077         /* XXX: make these macro values. */
2078         /* FIXME: warn about missing values. */
2079         mdoc->meta.title = mandoc_strdup("UNKNOWN");
2080         mdoc->meta.vol = mandoc_strdup("LOCAL");
2081         mdoc->meta.msec = mandoc_strdup("1");
2082         return(1);
2083     }

2085     /* Handles: '.Dt TITLE'
2086      * --> title = TITLE, volume = local, msec = 0, arch = NULL
2087      */

2089     mdoc->meta.title = mandoc_strdup
2090         ('\0' == nn->string[0] ? "UNKNOWN" : nn->string);

2092     if (NULL == (nn = nn->next)) {
2093         /* FIXME: warn about missing msec. */
2094         /* XXX: make this a macro value. */
2095         mdoc->meta.vol = mandoc_strdup("LOCAL");
2096         mdoc->meta.msec = mandoc_strdup("1");
2097         return(1);
2098     }

2100     /* Handles: '.Dt TITLE SEC'
2101      * --> title = TITLE, volume = SEC is msec ?
2102      * format(msec) : SEC,

```

```

2103 *      msec = SEC is msec ? atoi(msec) : 0,
2104 *      arch = NULL
2105 */

2107 cp = mdoc_a2msec(nn->string);
2108 if (cp) {
2109     mdoc->meta.vol = mdoc_strdup(cp);
2110     mdoc->meta.msec = mdoc_strdup(nn->string);
2111 } else {
2112     mdoc_nmsg(mdoc, n, MANDOCERR_BADMSEC);
2113     mdoc->meta.vol = mdoc_strdup(nn->string);
2114     mdoc->meta.msec = mdoc_strdup(nn->string);
2115 }

2117 if (NULL == (nn = nn->next))
2118     return(1);

2120 /* Handles: \.Dt TITLE SEC VOL'
2121 *  --> title = TITLE, volume = VOL is vol ?
2122 *      format(VOL) :
2123 *      VOL is arch ? format(arch) :
2124 *      VOL
2125 */

2127 cp = mdoc_a2vol(nn->string);
2128 if (cp) {
2129     free(mdoc->meta.vol);
2130     mdoc->meta.vol = mdoc_strdup(cp);
2131 } else {
2132     /* FIXME: warn about bad arch. */
2133     cp = mdoc_a2arch(nn->string);
2134     if (NULL == cp) {
2135         free(mdoc->meta.vol);
2136         mdoc->meta.vol = mdoc_strdup(nn->string);
2137     } else
2138         mdoc->meta.arch = mdoc_strdup(cp);
2139 }

2141 /* Ignore any subsequent parameters... */
2142 /* FIXME: warn about subsequent parameters. */

2144 return(1);
2145 }

2147 static int
2148 post_prol(POST_ARGS)
2149 {
2150     /*
2151     * Remove prologue macros from the document after they're
2152     * processed. The final document uses mdoc_meta for these
2153     * values and discards the originals.
2154     */

2156     mdoc_node_delete(mdoc, mdoc->last);
2157     if (mdoc->meta.title && mdoc->meta.date && mdoc->meta.os)
2158         mdoc->flags |= MDOC_PBODY;

2160     return(1);
2161 }

2163 static int
2164 post_bx(POST_ARGS)
2165 {
2166     struct mdoc_node      *n;
2168     /*

```

```

2169 * Make `Bx's second argument always start with an uppercase
2170 * letter. Groff checks if it's an "accepted" term, but we just
2171 * uppercase blindly.
2172 */

2174 n = mdoc->last->child;
2175 if (n && NULL != (n = n->next))
2176     *n->string = (char)toupper
2177         ((unsigned char)*n->string);

2179     return(1);
2180 }

2182 static int
2183 post_os(POST_ARGS)
2184 {
2185     struct mdoc_node *n;
2186     char             buf[BUFSIZ];
2187     int              c;
2188 #ifndef OSNAME
2189     struct utsname   utsname;
2190 #endif

2192     n = mdoc->last;

2194     /*
2195     * Set the operating system by way of the 'Os' macro. Note that
2196     * if an argument isn't provided and -DOSNAME="\foo\" is
2197     * provided during compilation, this value will be used instead
2198     * of filling in "sysname release" from uname().
2199     */

2201     if (mdoc->meta.os)
2202         free(mdoc->meta.os);

2204     buf[0] = '\0';
2205     if (-1 == (c = concat(buf, n->child, BUFSIZ))) {
2206         mdoc_nmsg(mdoc, n->child, MANDOCERR_MEM);
2207         return(0);
2208     }

2210     assert(c);

2212     /* XXX: yes, these can all be dynamically-adjusted buffers, but
2213     * it's really not worth the extra hackery.
2214     */

2216     if ('\0' == buf[0]) {
2217 #ifdef OSNAME
2218         if (strlcat(buf, OSNAME, BUFSIZ) >= BUFSIZ) {
2219             mdoc_nmsg(mdoc, n, MANDOCERR_MEM);
2220             return(0);
2221         }
2222 #else /*!OSNAME */
2223         if (-1 == uname(&utsname)) {
2224             mdoc_nmsg(mdoc, n, MANDOCERR_UNAME);
2225             mdoc->meta.os = mdoc_strdup("UNKNOWN");
2226             return(post_prol(mdoc));
2227         }

2229         if (strlcat(buf, utsname.sysname, BUFSIZ) >= BUFSIZ) {
2230             mdoc_nmsg(mdoc, n, MANDOCERR_MEM);
2231             return(0);
2232         }
2233         if (strlcat(buf, " ", BUFSIZ) >= BUFSIZ) {
2234             mdoc_nmsg(mdoc, n, MANDOCERR_MEM);

```

```

2235         return(0);
2236     }
2237     if (strlcat(buf, utsname.release, BUFSIZ) >= BUFSIZ) {
2238         mdoc_nmsg(mdoc, n, MANDOCERR_MEM);
2239         return(0);
2240     }
2241 #endif /*!OSNAME*/
2242 }

2244     mdoc->meta.os = mandoc_strdup(buf);
2245     return(1);
2246 }

2248 static int
2249 post_std(POST_ARGS)
2250 {
2251     struct mdoc_node *nn, *n;

2253     n = mdoc->last;

2255     /*
2256      * Macros accepting '-std' as an argument have the name of the
2257      * current document ('Nm') filled in as the argument if it's not
2258      * provided.
2259      */

2261     if (n->child)
2262         return(1);

2264     if (NULL == mdoc->meta.name)
2265         return(1);

2267     nn = n;
2268     mdoc->next = MDOC_NEXT_CHILD;

2270     if (! mdoc_word_alloc(mdoc, n->line, n->pos, mdoc->meta.name))
2271         return(0);

2273     mdoc->last = nn;
2274     return(1);
2275 }

2277 /*
2278  * Concatenate a node, stopping at the first non-text.
2279  * Concatenation is separated by a single whitespace.
2280  * Returns -1 on fatal (string overrun) error, 0 if child nodes were
2281  * encountered, 1 otherwise.
2282  */
2283 static int
2284 concat(char *p, const struct mdoc_node *n, size_t sz)
2285 {
2287     for ( ; NULL != n; n = n->next) {
2288         if (MDOC_TEXT != n->type)
2289             return(0);
2290         if ('\0' != p[0] && strlcat(p, " ", sz) >= sz)
2291             return(-1);
2292         if (strlcat(p, n->string, sz) >= sz)
2293             return(-1);
2294         concat(p, n->child, sz);
2295     }

2297     return(1);
2298 }

2300 static enum mdoc_sec

```

```

2301 a2sec(const char *p)
2302 {
2303     int         i;

2305     for (i = 0; i < (int)SEC_MAX; i++)
2306         if (secnames[i] && 0 == strcmp(p, secnames[i]))
2307             return((enum mdoc_sec)i);

2309     return(SEC_CUSTOM);
2310 }

2312 static size_t
2313 macro2len(enum mdoct macro)
2314 {
2316     switch (macro) {
2317     case(MDOC_Ad):
2318         return(12);
2319     case(MDOC_Ao):
2320         return(12);
2321     case(MDOC_An):
2322         return(12);
2323     case(MDOC_Aq):
2324         return(12);
2325     case(MDOC_Ar):
2326         return(12);
2327     case(MDOC_Bo):
2328         return(12);
2329     case(MDOC_Bq):
2330         return(12);
2331     case(MDOC_Cd):
2332         return(12);
2333     case(MDOC_Cm):
2334         return(10);
2335     case(MDOC_Do):
2336         return(10);
2337     case(MDOC_Dq):
2338         return(12);
2339     case(MDOC_Dv):
2340         return(12);
2341     case(MDOC_Eo):
2342         return(12);
2343     case(MDOC_Em):
2344         return(10);
2345     case(MDOC_Er):
2346         return(17);
2347     case(MDOC_Ev):
2348         return(15);
2349     case(MDOC_Fa):
2350         return(12);
2351     case(MDOC_Fl):
2352         return(10);
2353     case(MDOC_Fo):
2354         return(16);
2355     case(MDOC_Fn):
2356         return(16);
2357     case(MDOC_Ic):
2358         return(10);
2359     case(MDOC_Li):
2360         return(16);
2361     case(MDOC_Ms):
2362         return(6);
2363     case(MDOC_Nm):
2364         return(10);
2365     case(MDOC_No):
2366         return(12);

```



```
2367     case(MDOC_Oo):
2368         return(10);
2369     case(MDOC_Op):
2370         return(14);
2371     case(MDOC_Pa):
2372         return(32);
2373     case(MDOC_Pf):
2374         return(12);
2375     case(MDOC_Po):
2376         return(12);
2377     case(MDOC_Pq):
2378         return(12);
2379     case(MDOC_Ql):
2380         return(16);
2381     case(MDOC_Qo):
2382         return(12);
2383     case(MDOC_So):
2384         return(12);
2385     case(MDOC_Sq):
2386         return(12);
2387     case(MDOC_Sy):
2388         return(6);
2389     case(MDOC_Sx):
2390         return(16);
2391     case(MDOC_Tn):
2392         return(10);
2393     case(MDOC_Va):
2394         return(12);
2395     case(MDOC_Vt):
2396         return(12);
2397     case(MDOC_Xr):
2398         return(10);
2399     default:
2400         break;
2401 };
2402 return(0);
2403 }
```

new/usr/src/cmd/mandoc/msec.c

1

1127 Sat Jul 19 19:24:31 2014

new/usr/src/cmd/mandoc/msec.c

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /*      $Id: msec.c,v 1.10 2011/12/02 01:37:14 schwarze Exp $ */
2 /*
3  * Copyright (c) 2009 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <stdlib.h>
22 #include <string.h>
23
24 #include "mandoc.h"
25 #include "libmandoc.h"
26
27 #define LINE(x, y) \
28     if (0 == strcmp(p, x)) return(y);
29
30 const char *
31 mandoc_a2msec(const char *p)
32 {
33
34 #include "msec.in"
35
36     return(NULL);
37 }
```

11663 Sat Jul 19 19:24:31 2014

new/usr/src/cmd/mandoc/msec.in

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 /*
2  * This file and its contents are supplied under the terms of the
3  * Common Development and Distribution License ("CDDL"), version 1.0.
4  * You may only use this file in accordance with the terms of version
5  * 1.0 of the CDDL.
6  *
7  * A full copy of the text of the CDDL should have accompanied this
8  * source. A copy of the CDDL is also available via the Internet at
9  * http://www.illumos.org/license/CDDL.
10 */
11
12 /*
13  * Copyright 2012 Nexenta Systems, Inc. All rights reserved.
14  */
15
16 LINE("1", "User Commands")
17 LINE("1B", "illumos/BSD Compatibility Package Commands")
18 LINE("1b", "illumos/BSD Compatibility Package Commands")
19 LINE("1C", "Communication Commands")
20 LINE("1c", "Communication Commands")
21 LINE("1F", "FMI Commands")
22 LINE("1f", "FMI Commands")
23 LINE("1G", "Graphics and CAD Commands")
24 LINE("1g", "Graphics and CAD Commands")
25 LINE("1HAS", "User Commands")
26 LINE("1has", "User Commands")
27 LINE("1M", "Maintenance Commands")
28 LINE("1m", "Maintenance Commands")
29 LINE("1S", "illumos Specific Commands")
30 LINE("1s", "illumos Specific Commands")
31 LINE("2", "System Calls")
32 LINE("3", "Introduction to Library Functions")
33 LINE("3AIO", "Asynchronous I/O Library Functions")
34 LINE("3aio", "Asynchronous I/O Library Functions")
35 LINE("3BSM", "Security and Auditing Library Functions")
36 LINE("3bsm", "Security and Auditing Library Functions")
37 LINE("3C", "Standard C Library Functions")
38 LINE("3c", "Standard C Library Functions")
39 LINE("3C_DB", "Threads Debugging Library Functions")
40 LINE("3C_db", "Threads Debugging Library Functions")
41 LINE("3CFGADM", "Configuration Administration Library Functions")
42 LINE("3cfgadm", "Configuration Administration Library Functions")
43 LINE("3COMPPUTIL", "Communication Protocol Parser Utilities Library Functions")
44 LINE("3compputil", "Communication Protocol Parser Utilities Library Functions")
45 LINE("3CONTRACT", "Contract Management Library Functions")
46 LINE("3contract", "Contract Management Library Functions")
47 LINE("3CPC", "CPU Performance Counters Library Functions")
48 LINE("3cpc", "CPU Performance Counters Library Functions")
49 LINE("3CURSES", "Curses Library Functions")
50 LINE("3curses", "Curses Library Functions")
51 LINE("3DAT", "Direct Access Transport Library Functions")
52 LINE("3dat", "Direct Access Transport Library Functions")
53 LINE("3DEVID", "Device ID Library Functions")
54 LINE("3devvid", "Device ID Library Functions")
55 LINE("3DEVINFO", "Device Information Library Functions")
56 LINE("3devinfo", "Device Information Library Functions")

```

```

57 LINE("3DL", "Dynamic Linking Library Functions")
58 LINE("3dl", "Dynamic Linking Library Functions")
59 LINE("3DLPI", "Data Link Provider Interface Library Functions")
60 LINE("3dlpi", "Data Link Provider Interface Library Functions")
61 LINE("3DMI", "DMI Library Functions")
62 LINE("3dmi", "DMI Library Functions")
63 LINE("3DNS_SD", "DNS Service Discovery Library Functions")
64 LINE("3dns_sd", "DNS Service Discovery Library Functions")
65 LINE("3DOOR", "Door Library Functions")
66 LINE("3door", "Door Library Functions")
67 LINE("3ELF", "ELF Library Functions")
68 LINE("3elf", "ELF Library Functions")
69 LINE("3EXACCT", "Extended Accounting File Access Library Functions")
70 LINE("3exacct", "Extended Accounting File Access Library Functions")
71 LINE("3EXT", "Extended Library Functions")
72 LINE("3ext", "Extended Library Functions")
73 LINE("3FCOE", "FCoE Port Management Library Functions")
74 LINE("3fcoe", "FCoE Port Management Library Functions")
75 LINE("3FSTYP", "File System Type Identification Library Functions")
76 LINE("3fstyp", "File System Type Identification Library Functions")
77 LINE("3GEN", "String Pattern-Matching Library Functions")
78 LINE("3gen", "String Pattern-Matching Library Functions")
79 LINE("3GSS", "Generic Security Services API Library Functions")
80 LINE("3gss", "Generic Security Services API Library Functions")
81 LINE("3HEAD", "Headers")
82 LINE("3head", "Headers")
83 LINE("3ISCSIT", "iSCSI Management Library Functions")
84 LINE("3iscsit", "iSCSI Management Library Functions")
85 LINE("3KRB", "Kerberos Library Functions")
86 LINE("3krb", "Kerberos Library Functions")
87 LINE("3KSTAT", "Kernel Statistics Library Functions")
88 LINE("3kstat", "Kernel Statistics Library Functions")
89 LINE("3KVM", "Kernel VM Library Functions")
90 LINE("3kvm", "Kernel VM Library Functions")
91 LINE("3LDAP", "LDAP Library Functions")
92 LINE("3ldap", "LDAP Library Functions")
93 LINE("3LGRP", "Locality Group Library Functions")
94 LINE("3lgrp", "Locality Group Library Functions")
95 LINE("3LIB", "Interface Libraries")
96 LINE("3lib", "Interface Libraries")
97 LINE("3LIBUCB", "illumos/BSD Compatibility Interface Libraries")
98 LINE("3libucb", "illumos/BSD Compatibility Interface Libraries")
99 LINE("3M", "Mathematical Library Functions")
100 LINE("3m", "Mathematical Library Functions")
101 LINE("3MAIL", "User Mailbox Library Functions")
102 LINE("3mail", "User Mailbox Library Functions")
103 LINE("3MALLOC", "Memory Allocation Library Functions")
104 LINE("3malloc", "Memory Allocation Library Functions")
105 LINE("3MP", "Multiple Precision Library Functions")
106 LINE("3mp", "Multiple Precision Library Functions")
107 LINE("3MPAPI", "Common Multipath Management Library Functions")
108 LINE("3mpapi", "Common Multipath Management Library Functions")
109 LINE("3NSL", "Networking Services Library Functions")
110 LINE("3nsl", "Networking Services Library Functions")
111 LINE("3NVPAIR", "Name-value Pair Library Functions")
112 LINE("3nvpair", "Name-value Pair Library Functions")
113 LINE("3PAM", "PAM Library Functions")
114 LINE("3pam", "PAM Library Functions")
115 LINE("3PAPI", "PAPI Library Functions")
116 LINE("3papi", "PAPI Library Functions")
117 LINE("3PERL", "Perl Library Functions")
118 LINE("3perl", "Perl Library Functions")
119 LINE("3PICL", "PICL Library Functions")
120 LINE("3picl", "PICL Library Functions")
121 LINE("3PICLTREE", "PICL Plug-In Library Functions")
122 LINE("3picltree", "PICL Plug-In Library Functions")

```

```

123 LINE("3PLOT", "Graphics Interface Library Functions")
124 LINE("3plot", "Graphics Interface Library Functions")
125 LINE("3POOL", "Pool Configuration Manipulation Library Functions")
126 LINE("3pool", "Pool Configuration Manipulation Library Functions")
127 LINE("3PROC", "Process Control Library Functions")
128 LINE("3proc", "Process Control Library Functions")
129 LINE("3PROJECT", "Project Database Access Library Functions")
130 LINE("3project", "Project Database Access Library Functions")
131 LINE("3RAC", "Remote Asynchronous Calls Library Functions")
132 LINE("3rac", "Remote Asynchronous Calls Library Functions")
133 LINE("3RESOLV", "Resolver Library Functions")
134 LINE("3resolv", "Resolver Library Functions")
135 LINE("3RPC", "RPC Library Functions")
136 LINE("3rpc", "RPC Library Functions")
137 LINE("3RSM", "Remote Shared Memory Library Functions")
138 LINE("3rsm", "Remote Shared Memory Library Functions")
139 LINE("3RT", "Realtime Library Functions")
140 LINE("3rt", "Realtime Library Functions")
141 LINE("3SASL", "Simple Authentication Security Layer Library Functions")
142 LINE("3sas1", "Simple Authentication Security Layer Library Functions")
143 LINE("3SCF", "Service Configuration Facility Library Functions")
144 LINE("3scf", "Service Configuration Facility Library Functions")
145 LINE("3SCHED", "LWP Scheduling Library Functions")
146 LINE("3sched", "LWP Scheduling Library Functions")
147 LINE("3SEC", "File Access Control Library Functions")
148 LINE("3sec", "File Access Control Library Functions")
149 LINE("3SECDB", "Security Attributes Database Library Functions")
150 LINE("3secdb", "Security Attributes Database Library Functions")
151 LINE("3SIP", "Session Initiation Protocol Library Functions")
152 LINE("3sip", "Session Initiation Protocol Library Functions")
153 LINE("3SLP", "Service Location Protocol Library Functions")
154 LINE("3slp", "Service Location Protocol Library Functions")
155 LINE("3SNMP", "SNMP Library Functions")
156 LINE("3snmp", "SNMP Library Functions")
157 LINE("3SOCKET", "Sockets Library Functions")
158 LINE("3socket", "Sockets Library Functions")
159 LINE("3STMF", "SCSI Target Mode Framework Library Functions")
160 LINE("3stmf", "SCSI Target Mode Framework Library Functions")
161 LINE("3SYSEVENT", "System Event Library Functions")
162 LINE("3ysevent", "System Event Library Functions")
163 LINE("3TECLA", "Interactive Command-line Input Library Functions")
164 LINE("3tecla", "Interactive Command-line Input Library Functions")
165 LINE("3THR", "Threads Library Functions")
166 LINE("3thr", "Threads Library Functions")
167 LINE("3TNF", "TNF Library Functions")
168 LINE("3tnf", "TNF Library Functions")
169 LINE("3TSOL", "Trusted Extensions Library Functions")
170 LINE("3tsol", "Trusted Extensions Library Functions")
171 LINE("3UCB", "illumos/BSD Compatibility Library Functions")
172 LINE("3ucb", "illumos/BSD Compatibility Library Functions")
173 LINE("3UUID", "Universally Unique Identifier Library Functions")
174 LINE("3uuid", "Universally Unique Identifier Library Functions")
175 LINE("3VOLMGT", "Volume Management Library Functions")
176 LINE("3volmgt", "Volume Management Library Functions")
177 LINE("3XCURSES", "X/Open Curses Library Functions")
178 LINE("3xcurses", "X/Open Curses Library Functions")
179 LINE("3XFN", "XFN Interface Library Functions")
180 LINE("3xfn", "XFN Interface Library Functions")
181 LINE("3XNET", "X/Open Networking Services Library Functions")
182 LINE("3xnet", "X/Open Networking Services Library Functions")
183 LINE("3B", "illumos/BSD Compatibility Library Functions")
184 LINE("3b", "illumos/BSD Compatibility Library Functions")
185 LINE("3C", "C Library Functions")
186 LINE("3c", "C Library Functions")
187 LINE("3F", "Fortran Library Routines")
188 LINE("3f", "Fortran Library Routines")

```

```

189 LINE("3G", "C Library Functions")
190 LINE("3g", "C Library Functions")
191 LINE("3K", "Kernel VM Library Functions")
192 LINE("3k", "Kernel VM Library Functions")
193 LINE("3L", "Lightweight Processes Library")
194 LINE("3l", "Lightweight Processes Library")
195 LINE("3N", "Network Functions")
196 LINE("3n", "Network Functions")
197 LINE("3R", "Realtime Library")
198 LINE("3r", "Realtime Library")
199 LINE("3S", "Standard I/O Functions")
200 LINE("3s", "Standard I/O Functions")
201 LINE("3T", "Threads Library")
202 LINE("3t", "Threads Library")
203 LINE("3W", "C Library Functions")
204 LINE("3w", "C Library Functions")
205 LINE("3X", "Miscellaneous Library Functions")
206 LINE("3x", "Miscellaneous Library Functions")
207 LINE("3XC", "X/Open Curses Library Functions")
208 LINE("3xc", "X/Open Curses Library Functions")
209 LINE("3XN", "X/Open Networking Services Library Functions")
210 LINE("3xn", "X/Open Networking Services Library Functions")
211 LINE("4", "File Formats")
212 LINE("4B", "illumos/BSD Compatibility Package File Formats")
213 LINE("4b", "illumos/BSD Compatibility Package File Formats")
214 LINE("5", "Standards, Environments, and Macros")
215 LINE("6", "Games and Demos")
216 LINE("7", "Device and Network Interfaces")
217 LINE("7B", "illumos/BSD Compatibility Special Files")
218 LINE("7b", "illumos/BSD Compatibility Special Files")
219 LINE("7D", "Devices")
220 LINE("7d", "Devices")
221 LINE("7FS", "File Systems")
222 LINE("7fs", "File Systems")
223 LINE("7I", "Ioctl Requests")
224 LINE("7i", "Ioctl Requests")
225 LINE("7IPP", "IP Quality of Service Modules")
226 LINE("7ipp", "IP Quality of Service Modules")
227 LINE("7M", "STREAMS Modules")
228 LINE("7m", "STREAMS Modules")
229 LINE("7P", "Protocols")
230 LINE("7p", "Protocols")
231 LINE("8", "Maintenance Procedures")
232 LINE("8C", "Maintenance Procedures")
233 LINE("8c", "Maintenance Procedures")
234 LINE("8S", "Maintenance Procedures")
235 LINE("8s", "Maintenance Procedures")
236 LINE("9", "Device Driver Interfaces")
237 LINE("9E", "Driver Entry Points")
238 LINE("9e", "Driver Entry Points")
239 LINE("9F", "Kernel Functions for Drivers")
240 LINE("9f", "Kernel Functions for Drivers")
241 LINE("9P", "Kernel Properties for Drivers")
242 LINE("9p", "Kernel Properties for Drivers")
243 LINE("9S", "Data Structures for Drivers")
244 LINE("9s", "Data Structures for Drivers")

```

```

*****
6614 Sat Jul 19 19:24:31 2014
new/usr/src/cmd/mandoc/out.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: out.c,v 1.43 2011/09/20 23:05:49 schwarze Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <sys/types.h>
23
24 #include <assert.h>
25 #include <ctype.h>
26 #include <stdio.h>
27 #include <stdlib.h>
28 #include <string.h>
29 #include <time.h>
30
31 #include "mandoc.h"
32 #include "out.h"
33
34 static void  tblcalc_data(struct rofftbl *, struct roffcol *,
35                       const struct tbl *, const struct tbl_dat *);
36 static void  tblcalc_literal(struct rofftbl *, struct roffcol *,
37                             const struct tbl_dat *);
38 static void  tblcalc_number(struct rofftbl *, struct roffcol *,
39                             const struct tbl *, const struct tbl_dat *);
40
41 /*
42  * Convert a 'scaling unit' to a consistent form, or fail.  Scaling
43  * units are documented in groff.7, mdoc.7, man.7.
44  */
45 int
46 a2roffsu(const char *src, struct roffsu *dst, enum roffscale def)
47 {
48     char      buf[BUFSIZ], hasd;
49     int       i;
50     enum roffscale  unit;
51
52     if ('\0' == *src)
53         return(0);
54
55     i = hasd = 0;

```

```

57     switch (*src) {
58     case ('+'):
59         src++;
60         break;
61     case ('-'):
62         buf[i++] = *src++;
63         break;
64     default:
65         break;
66     }
67
68     if ('\0' == *src)
69         return(0);
70
71     while (i < BUFSIZ) {
72         if (! isdigit((unsigned char)*src)) {
73             if ('.' != *src)
74                 break;
75             else if (hasd)
76                 break;
77             else
78                 hasd = 1;
79         }
80         buf[i++] = *src++;
81     }
82
83     if (BUFSIZ == i || (*src && *(src + 1)))
84         return(0);
85
86     buf[i] = '\0';
87
88     switch (*src) {
89     case ('c'):
90         unit = SCALE_CM;
91         break;
92     case ('i'):
93         unit = SCALE_IN;
94         break;
95     case ('P'):
96         unit = SCALE_PC;
97         break;
98     case ('p'):
99         unit = SCALE_PT;
100        break;
101     case ('f'):
102         unit = SCALE_FS;
103         break;
104     case ('v'):
105         unit = SCALE_VS;
106         break;
107     case ('m'):
108         unit = SCALE_EM;
109         break;
110     case ('\0'):
111         if (SCALE_MAX == def)
112             return(0);
113         unit = SCALE_BU;
114         break;
115     case ('u'):
116         unit = SCALE_BU;
117         break;
118     case ('M'):
119         unit = SCALE_MM;
120         break;
121     case ('n'):
122         unit = SCALE_EN;

```

```

123         break;
124     default:
125         return(0);
126     }

128     /* FIXME: do this in the caller. */
129     if ((dst->scale = atof(buf)) < 0)
130         dst->scale = 0;
131     dst->unit = unit;
132     return(1);
133 }

135 /*
136  * Calculate the abstract widths and decimal positions of columns in a
137  * table. This routine allocates the columns structures then runs over
138  * all rows and cells in the table. The function pointers in "tbl" are
139  * used for the actual width calculations.
140  */
141 void
142 tblcalc(struct rofftbl *tbl, const struct tbl_span *sp)
143 {
144     const struct tbl_dat *dp;
145     const struct tbl_head *hp;
146     struct roffcol *col;
147     int spans;

149     /*
150      * Allocate the master column specifiers. These will hold the
151      * widths and decimal positions for all cells in the column. It
152      * must be freed and nullified by the caller.
153      */

155     assert(NULL == tbl->cols);
156     tbl->cols = mandoc_calloc
157         ((size_t)sp->tbl->cols, sizeof(struct roffcol));

159     hp = sp->head;

161     for ( ; sp; sp = sp->next) {
162         if (TBL_SPAN_DATA != sp->pos)
163             continue;
164         spans = 1;
165         /*
166          * Account for the data cells in the layout, matching it
167          * to data cells in the data section.
168          */
169         for (dp = sp->first; dp; dp = dp->next) {
170             /* Do not used spanned cells in the calculation. */
171             if (0 < --spans)
172                 continue;
173             spans = dp->spans;
174             if (1 < spans)
175                 continue;
176             assert(dp->layout);
177             col = &tbl->cols[dp->layout->head->ident];
178             tblcalc_data(tbl, col, sp->tbl, dp);
179         }
180     }

182     /*
183      * Calculate width of the spanners. These get one space for a
184      * vertical line, two for a double-vertical line.
185      */

187     for ( ; hp; hp = hp->next) {
188         col = &tbl->cols[hp->ident];

```

```

189         switch (hp->pos) {
190             case (TBL_HEAD_VERT):
191                 col->width = (*tbl->len)(1, tbl->arg);
192                 break;
193             case (TBL_HEAD_DVERT):
194                 col->width = (*tbl->len)(2, tbl->arg);
195                 break;
196             default:
197                 break;
198         }
199     }
200 }

202 static void
203 tblcalc_data(struct rofftbl *tbl, struct roffcol *col,
204             const struct tbl *tp, const struct tbl_dat *dp)
205 {
206     size_t sz;

208     /* Branch down into data sub-types. */

210     switch (dp->layout->pos) {
211         case (TBL_CELL_HORIZ):
212             /* FALLTHROUGH */
213         case (TBL_CELL_DHORIZ):
214             sz = (*tbl->len)(1, tbl->arg);
215             if (col->width < sz)
216                 col->width = sz;
217             break;
218         case (TBL_CELL_LONG):
219             /* FALLTHROUGH */
220         case (TBL_CELL_CENTRE):
221             /* FALLTHROUGH */
222         case (TBL_CELL_LEFT):
223             /* FALLTHROUGH */
224         case (TBL_CELL_RIGHT):
225             tblcalc_literal(tbl, col, dp);
226             break;
227         case (TBL_CELL_NUMBER):
228             tblcalc_number(tbl, col, tp, dp);
229             break;
230         case (TBL_CELL_DOWN):
231             break;
232         default:
233             abort();
234             /* NOTREACHED */
235     }
236 }

238 static void
239 tblcalc_literal(struct rofftbl *tbl, struct roffcol *col,
240             const struct tbl_dat *dp)
241 {
242     size_t sz;
243     const char *str;

245     str = dp->string ? dp->string : "";
246     sz = (*tbl->slen)(str, tbl->arg);

248     if (col->width < sz)
249         col->width = sz;
250 }

252 static void
253 tblcalc_number(struct rofftbl *tbl, struct roffcol *col,
254             const struct tbl *tp, const struct tbl_dat *dp)

```

```
255 {
256     int            i;
257     size_t        sz, psz, ssz, d;
258     const char    *str;
259     char          *cp;
260     char          buf[2];
261
262     /*
263     * First calculate number width and decimal place (last + 1 for
264     * non-decimal numbers).  If the stored decimal is subsequent to
265     * ours, make our size longer by that difference
266     * (right-"shifting"); similarly, if ours is subsequent the
267     * stored, then extend the stored size by the difference.
268     * Finally, re-assign the stored values.
269     */
270
271     str = dp->string ? dp->string : "";
272     sz = (*tbl->slen)(str, tbl->arg);
273
274     /* FIXME: TBL_DATA_HORIZ et al.? */
275
276     buf[0] = tp->decimal;
277     buf[1] = '\0';
278
279     psz = (*tbl->slen)(buf, tbl->arg);
280
281     if (NULL != (cp = strrchr(str, tp->decimal))) {
282         buf[1] = '\0';
283         for (ssz = 0, i = 0; cp != &str[i]; i++) {
284             buf[0] = str[i];
285             ssz += (*tbl->slen)(buf, tbl->arg);
286         }
287         d = ssz + psz;
288     } else
289         d = sz + psz;
290
291     /* Adjust the settings for this column. */
292
293     if (col->decimal > d) {
294         sz += col->decimal - d;
295         d = col->decimal;
296     } else
297         col->width += d - col->decimal;
298
299     if (sz > col->width)
300         col->width = sz;
301     if (d > col->decimal)
302         col->decimal = d;
303 }
```

new/usr/src/cmd/mandoc/out.h

1

2124 Sat Jul 19 19:24:31 2014

new/usr/src/cmd/mandoc/out.h

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /* $Id: out.h,v 1.21 2011/07/17 15:24:25 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef OUT_H
18 #define OUT_H
19
20 enum roffscale {
21     SCALE_CM, /* centimeters (c) */
22     SCALE_IN, /* inches (i) */
23     SCALE_PC, /* pica (P) */
24     SCALE_PT, /* points (p) */
25     SCALE_EM, /* ems (m) */
26     SCALE_MM, /* mini-ems (M) */
27     SCALE_EN, /* ens (n) */
28     SCALE_BU, /* default horizontal (u) */
29     SCALE_VS, /* default vertical (v) */
30     SCALE_FS, /* syn. for u (f) */
31     SCALE_MAX
32 };
33
34 struct roffcol {
35     size_t width; /* width of cell */
36     size_t decimal; /* decimal position in cell */
37 };
38
39 struct roffsu {
40     enum roffscale unit;
41     double scale;
42 };
43
44 typedef size_t (*tbl_strlen)(const char *, void *);
45 typedef size_t (*tbl_len)(size_t, void *);
46
47 struct rofftbl {
48     tbl_strlen slen; /* calculate string length */
49     tbl_len len; /* produce width of empty space */
50     struct roffcol *cols; /* master column specifiers */
51     void *arg; /* passed to slen and len */
52 };
53
54 __BEGIN_DECLS
55
56 #define SCALE_VS_INIT(p, v) \
```

new/usr/src/cmd/mandoc/out.h

2

```
57     do { (p)->unit = SCALE_VS; \
58         (p)->scale = (v); } \
59     while (/* CONSTCOND */ 0)
60
61 #define SCALE_HS_INIT(p, v) \
62     do { (p)->unit = SCALE_BU; \
63         (p)->scale = (v); } \
64     while (/* CONSTCOND */ 0)
65
66 int a2roffsu(const char *, struct roffsu *, enum roffscale);
67 void tblcalc(struct rofftbl *tbl, const struct tbl_span *);
68
69 __END_DECLS
70
71 #endif /* !OUT_H */
```



```

*****
10389 Sat Jul 19 19:24:31 2014
new/usr/src/cmd/mandoc/preconv.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*
2  * $Id: preconv.c,v 1.5 2011/07/24 18:15:14 kristaps Exp $ */
3  * Copyright (c) 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #ifdef HAVE_MMAP
22 #include <sys/stat.h>
23 #include <sys/mman.h>
24 #endif
25
26 #include <assert.h>
27 #include <fcntl.h>
28 #include <stdio.h>
29 #include <stdlib.h>
30 #include <string.h>
31 #include <unistd.h>
32
33 /*
34  * The read_whole_file() and resize_buf() functions are copied from
35  * read.c, including all dependency code (MAP_FILE, etc.).
36  */
37
38 #ifndef MAP_FILE
39 #define MAP_FILE 0
40 #endif
41
42 enum enc {
43     ENC_UTF_8, /* UTF-8 */
44     ENC_US_ASCII, /* US-ASCII */
45     ENC_LATIN_1, /* Latin-1 */
46     ENC_MAX
47 };
48
49 struct buf {
50     char      *buf; /* binary input buffer */
51     size_t    sz; /* size of binary buffer */
52     size_t    ofs; /* starting buffer offset */
53 };
54
55 struct encode {
56     const char *name;

```

```

57     int      (*conv)(const struct buf *);
58 };
59
60 static int cue_enc(const struct buf *, size_t *, enum enc *);
61 static int conv_latin_1(const struct buf *);
62 static int conv_us_ascii(const struct buf *);
63 static int conv_utf_8(const struct buf *);
64 static int read_whole_file(const char *, int,
65     struct buf *, int *);
66 static void resize_buf(struct buf *, size_t);
67 static void usage(void);
68
69 static const struct encode encs[ENC_MAX] = {
70     { "utf-8", conv_utf_8 }, /* ENC_UTF_8 */
71     { "us-ascii", conv_us_ascii }, /* ENC_US_ASCII */
72     { "latin-1", conv_latin_1 }, /* ENC_LATIN_1 */
73 };
74
75 static const char *progname;
76
77 static void
78 usage(void)
79 {
80     fprintf(stderr, "usage: %s "
81         "[-D enc] "
82         "[-e ENC] "
83         "[file]\n", progname);
84 }
85
86
87 static int
88 conv_latin_1(const struct buf *b)
89 {
90     size_t i;
91     unsigned char cu;
92     const char *cp;
93
94     cp = b->buf + (int)b->ofs;
95
96     /*
97      * Latin-1 falls into the first 256 code-points of Unicode, so
98      * there's no need for any sort of translation. Just make the
99      * 8-bit characters use the Unicode escape.
100     * Note that binary values 128 < v < 160 are passed through
101     * unmodified to mandoc.
102     */
103
104     for (i = b->ofs; i < b->sz; i++) {
105         cu = (unsigned char)*cp++;
106         cu < 160U ? putchar(cu) : printf("\\[u%.4X]", cu);
107     }
108
109     return(1);
110 }
111
112 static int
113 conv_us_ascii(const struct buf *b)
114 {
115     /*
116      * US-ASCII has no conversion since it falls into the first 128
117      * bytes of Unicode.
118      */
119
120     fwrite(b->buf, 1, b->sz, stdout);
121     return(1);

```

```

123 }
125 static int
126 conv_utf_8(const struct buf *b)
127 {
128     int             state, be;
129     unsigned int    accum;
130     size_t          i;
131     unsigned char   cu;
132     const char      *cp;
133     const long      one = 1L;
134
135     cp = b->buf + (int)b->offs;
136     state = 0;
137     accum = 0U;
138     be = 0;
139
140     /* Quick test for big-endian value. */
141
142     if ( ! (*(const char *)&one) )
143         be = 1;
144
145     for (i = b->offs; i < b->sz; i++) {
146         cu = (unsigned char)*cp++;
147         if (state) {
148             if ( ! (cu & 128) || (cu & 64) ) {
149                 /* Bad sequence header. */
150                 return(0);
151             }
152
153             /* Accept only legitimate bit patterns. */
154
155             if (cu > 191 || cu < 128) {
156                 /* Bad in-sequence bits. */
157                 return(0);
158             }
159
160             accum |= (cu & 63) << --state * 6;
161
162             /*
163              * Accum is held in little-endian order as
164              * stipulated by the UTF-8 sequence coding. We
165              * need to convert to a native big-endian if our
166              * architecture requires it.
167              */
168
169             if (0 == state && be)
170                 accum = (accum >> 24) |
171                     ((accum << 8) & 0x00FF0000) |
172                     ((accum >> 8) & 0x0000FF00) |
173                     (accum << 24);
174
175             if (0 == state) {
176                 accum < 128U ? putchar(accum) :
177                 printf("\\[u%.4X]", accum);
178                 accum = 0U;
179             }
180             } else if (cu & (1 << 7)) {
181                 /*
182                  * Entering a UTF-8 state: if we encounter a
183                  * UTF-8 bitmask, calculate the expected UTF-8
184                  * state from it.
185                  */
186                 for (state = 0; state < 7; state++)
187                     if ( ! (cu & (1 << (7 - state))))
188                         break;

```

```

190             /* Accept only legitimate bit patterns. */
191
192             switch (state) {
193                 case (4):
194                     if (cu <= 244 && cu >= 240) {
195                         accum = (cu & 7) << 18;
196                         break;
197                     }
198                     /* Bad 4-sequence start bits. */
199                     return(0);
200                 case (3):
201                     if (cu <= 239 && cu >= 224) {
202                         accum = (cu & 15) << 12;
203                         break;
204                     }
205                     /* Bad 3-sequence start bits. */
206                     return(0);
207                 case (2):
208                     if (cu <= 223 && cu >= 194) {
209                         accum = (cu & 31) << 6;
210                         break;
211                     }
212                     /* Bad 2-sequence start bits. */
213                     return(0);
214                 default:
215                     /* Bad sequence bit mask. */
216                     return(0);
217             }
218             state--;
219         } else
220             putchar(cu);
221     }
222
223     if (0 != state) {
224         /* Bad trailing bits. */
225         return(0);
226     }
227
228     return(1);
229 }
230
231 static void
232 resize_buf(struct buf *buf, size_t initial)
233 {
234
235     buf->sz = buf->sz > initial / 2 ?
236         2 * buf->sz : initial;
237
238     buf->buf = realloc(buf->buf, buf->sz);
239     if (NULL == buf->buf) {
240         perror(NULL);
241         exit(EXIT_FAILURE);
242     }
243 }
244
245 static int
246 read_whole_file(const char *f, int fd,
247                 struct buf *fb, int *with_mmap)
248 {
249     size_t          off;
250     ssize_t         ssz;
251
252     #ifdef HAVE_MMAP
253     struct stat     st;
254     if (-1 == fstat(fd, &st)) {

```

```

255         perror(f);
256         return(0);
257     }

259     /*
260     * If we're a regular file, try just reading in the whole entry
261     * via mmap(). This is faster than reading it into blocks, and
262     * since each file is only a few bytes to begin with, I'm not
263     * concerned that this is going to tank any machines.
264     */

266     if (S_ISREG(st.st_mode) && st.st_size >= (1U << 31)) {
267         fprintf(stderr, "%s: input too large\n", f);
268         return(0);
269     }

270     if (S_ISREG(st.st_mode)) {
271         *with_mmap = 1;
272         fb->sz = (size_t)st.st_size;
273         fb->buf = mmap(NULL, fb->sz, PROT_READ,
274                       MAP_FILE|MAP_SHARED, fd, 0);
275         if (fb->buf != MAP_FAILED)
276             return(1);
277     }
278 #endif
279

281     /*
282     * If this isn't a regular file (like, say, stdin), then we must
283     * go the old way and just read things in bit by bit.
284     */

286     *with_mmap = 0;
287     off = 0;
288     fb->sz = 0;
289     fb->buf = NULL;
290     for (;;) {
291         if (off == fb->sz && fb->sz == (1U << 31)) {
292             fprintf(stderr, "%s: input too large\n", f);
293             break;
294         }
295         if (off == fb->sz)
296             resize_buf(fb, 65536);
297
299         ssz = read(fd, fb->buf + (int)off, fb->sz - off);
300         if (ssz == 0) {
301             fb->sz = off;
302             return(1);
303         }
304         if (ssz == -1) {
305             perror(f);
306             break;
307         }
308         off += (size_t)ssz;
309     }

311     free(fb->buf);
312     fb->buf = NULL;
313     return(0);
314 }

316 static int
317 cue_enc(const struct buf *b, size_t *offs, enum enc *enc)
318 {
319     const char      *ln, *eoln, *eoph;
320     size_t          sz, phsz, nsz;

```

```

321     int            i;

323     ln = b->buf + (int)*offs;
324     sz = b->sz - *offs;

326     /* Look for the end-of-line. */

328     if (NULL == (eoln = memchr(ln, '\n', sz)))
329         return(-1);

331     /* Set next-line marker. */

333     *offs = (size_t)((eoln + 1) - b->buf);

335     /* Check if we have the correct header/trailer. */

337     if ((sz = (size_t)(eoln - ln)) < 10 ||
338         memcmp(ln, ".\\\\" "-*-", 7) ||
339         memcmp(eoln - 3, "-*-", 3))
340         return(0);

342     /* Move after the header and adjust for the trailer. */

344     ln += 7;
345     sz -= 10;

347     while (sz > 0) {
348         while (sz > 0 && ' ' == *ln) {
349             ln++;
350             sz--;
351         }
352         if (0 == sz)
353             break;

355         /* Find the end-of-phrase marker (or eoln). */

357         if (NULL == (eoph = memchr(ln, ';', sz)))
358             eoph = eoln - 3;
359         else
360             eoph++;

362         /* Only account for the "coding" phrase. */

364         if ((phsz = (size_t)(eoph - ln)) < 7 ||
365             strncasecmp(ln, "coding:", 7)) {
366             sz -= phsz;
367             ln += phsz;
368             continue;
369         }

371         sz -= 7;
372         ln += 7;

374         while (sz > 0 && ' ' == *ln) {
375             ln++;
376             sz--;
377         }
378         if (0 == sz)
379             break;

381         /* Check us against known encodings. */

383         for (i = 0; i < (int)ENC_MAX; i++) {
384             nsz = strlen(encs[i].name);
385             if (phsz < nsz)
386                 continue;

```

```

387         if (strncasecmp(ln, encs[i].name, nsz))
388             continue;
390         *enc = (enum enc)i;
391         return(1);
392     }
394     /* Unknown encoding. */
396     *enc = ENC__MAX;
397     return(1);
398 }
400 return(0);
401 }
403 int
404 main(int argc, char *argv[])
405 {
406     int            i, ch, map, fd, rc;
407     struct buf     b;
408     const char    *fn;
409     enum enc       enc, def;
410     unsigned char bom[3] = { 0xEF, 0xBB, 0xBF };
411     size_t        offs;
412     extern int     optind;
413     extern char    *optarg;
415     progname = strrchr(argv[0], '/');
416     if (progname == NULL)
417         progname = argv[0];
418     else
419         ++progname;
421     fn = "<stdin>";
422     fd = STDIN_FILENO;
423     rc = EXIT_FAILURE;
424     enc = def = ENC__MAX;
425     map = 0;
427     memset(&b, 0, sizeof(struct buf));
429     while (-1 != (ch = getopt(argc, argv, "D:e:rdvh")))
430         switch (ch) {
431             case ('D'):
432                 /* FALLTHROUGH */
433             case ('e'):
434                 for (i = 0; i < (int)ENC__MAX; i++) {
435                     if (strncasecmp(optarg, encs[i].name))
436                         continue;
437                     break;
438                 }
439                 if (i < (int)ENC__MAX) {
440                     if ('D' == ch)
441                         def = (enum enc)i;
442                     else
443                         enc = (enum enc)i;
444                     break;
445                 }
447                 fprintf(stderr, "%s: Bad encoding\n", optarg);
448                 return(EXIT_FAILURE);
449             case ('r'):
450                 /* FALLTHROUGH */
451             case ('d'):
452                 /* FALLTHROUGH */

```

```

453         case ('v'):
454             /* Compatibility with GNU preconv. */
455             break;
456         case ('h'):
457             /* Compatibility with GNU preconv. */
458             /* FALLTHROUGH */
459         default:
460             usage();
461             return(EXIT_FAILURE);
462     }
464     argc -= optind;
465     argv += optind;
466
467     /*
468     * Open and read the first argument on the command-line.
469     * If we don't have one, we default to stdin.
470     */
472     if (argc > 0) {
473         fn = *argv;
474         fd = open(fn, O_RDONLY, 0);
475         if (-1 == fd) {
476             perror(fn);
477             return(EXIT_FAILURE);
478         }
479     }
481     if (! read_whole_file(fn, fd, &b, &map))
482         goto out;
484     /* Try to read the UTF-8 BOM. */
486     if (ENC__MAX == enc)
487         if (b.sz > 3 && 0 == memcmp(b.buf, bom, 3)) {
488             b.offs = 3;
489             enc = ENC_UTF_8;
490         }
492     /* Try reading from the "-*--" cue. */
494     if (ENC__MAX == enc) {
495         offs = b.offs;
496         ch = cue_enc(&b, &offs, &enc);
497         if (0 == ch)
498             ch = cue_enc(&b, &offs, &enc);
499     }
501     /*
502     * No encoding has been detected.
503     * Thus, we either fall into our default encoder, if specified,
504     * or use Latin-1 if all else fails.
505     */
507     if (ENC__MAX == enc)
508         enc = ENC__MAX == def ? ENC_LATIN_1 : def;
510     if (! (*encs[(int)enc].conv)(&b)) {
511         fprintf(stderr, "%s: Bad encoding\n", fn);
512         goto out;
513     }
515     rc = EXIT_SUCCESS;
516 out:
517 #ifdef HAVE_MMAP
518     if (map)

```

```
519         munmap(b.buf, b.sz);
520     else
521 #endif
522         free(b.buf);
524     if (fd > STDIN_FILENO)
525         close(fd);
527     return(rc);
528 }
```

new/usr/src/cmd/mandoc/predefs.in

1

2091 Sat Jul 19 19:24:31 2014

new/usr/src/cmd/mandoc/predefs.in

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

1 /* \$Id: predefs.in,v 1.3 2011/07/31 11:36:49 schwarze Exp \$ */

2 /*

3 * Copyright (c) 2011 Kristaps Dzonsons <kristaps@bsd.lv>

4 *

5 * Permission to use, copy, modify, and distribute this software for any

6 * purpose with or without fee is hereby granted, provided that the above

7 * copyright notice and this permission notice appear in all copies.

8 *

9 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES

10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF

11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR

12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES

13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN

14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF

15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

16 */

18 /*

19 * The predefined-string translation tables. Each corresponds to a

20 * predefined strings from (e.g.) tmac/mdoc/doc-nroff. The left-hand

21 * side corresponds to the input sequence (*x, *(xx and so on). The

22 * right-hand side is what's produced by libroff.

23 *

24 * XXX - C-escape strings!

25 * XXX - update PREDEF_MAX in roff.c if adding more!

26 */

28 PREDEF("Am", "&")

29 PREDEF("Ba", "|")

30 PREDEF("Ge", "\(>=")

31 PREDEF("Gt", ">")

32 PREDEF("If", "infinity")

33 PREDEF("Le", "\(<=")

34 PREDEF("Lq", "\(<lq")

35 PREDEF("Lt", "<")

36 PREDEF("Na", "NaN")

37 PREDEF("Ne", "\(!=")

38 PREDEF("Pi", "pi")

39 PREDEF("Pm", "\(&+-")

40 PREDEF("Rq", "\(&rq")

41 PREDEF("left-bracket", "[")

42 PREDEF("left-parenthesis", "(")

43 PREDEF("lp", "(")

44 PREDEF("left-singlequote", "\(&oq")

45 PREDEF("q", "\(&dq")

46 PREDEF("quote-left", "\(&oq")

47 PREDEF("quote-right", "\(&cq")

48 PREDEF("R", "\(&rg")

49 PREDEF("right-bracket", "]")

50 PREDEF("right-parenthesis", ")")

51 PREDEF("rp", ")")

52 PREDEF("right-singlequote", "\(&cq")

53 PREDEF("Tm", "(Tm)")

54 PREDEF("Px", "POSIX")

55 PREDEF("Ai", "ANSI")

56 PREDEF("\', "\(&'")

new/usr/src/cmd/mandoc/predefs.in

2

57 PREDEF("aa", "\(&aa")

58 PREDEF("ga", "\(&ga")

59 PREDEF("\', "\(&'")

60 PREDEF("lq", "\(<lq")

61 PREDEF("rq", "\(<rq")

62 PREDEF("ua", "\(<ua")

63 PREDEF("va", "\(<va")

64 PREDEF("<=", "\(<=&")

65 PREDEF(">=", "\(>=&")

```

*****
19263 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/read.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: read.c,v 1.28 2012/02/16 20:51:31 joerg Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2010, 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #ifdef HAVE_MMAP
23 #include <sys/stat.h>
24 #include <sys/mman.h>
25 #endif
26
27 #include <assert.h>
28 #include <ctype.h>
29 #include <fcntl.h>
30 #include <stdarg.h>
31 #include <stdint.h>
32 #include <stdio.h>
33 #include <stdlib.h>
34 #include <string.h>
35 #include <unistd.h>
36
37 #include "mandoc.h"
38 #include "libmandoc.h"
39 #include "mdoc.h"
40 #include "man.h"
41 #include "main.h"
42
43 #ifndef MAP_FILE
44 #define MAP_FILE      0
45 #endif
46
47 #define REPARSE_LIMIT  1000
48
49 struct buf {
50     char          *buf; /* binary input buffer */
51     size_t        sz; /* size of binary buffer */
52 };
53
54 struct mparse {
55     enum mandoclevel file_status; /* status of current parse */
56     enum mandoclevel wlevel; /* ignore messages below this */

```

```

57     int          line; /* line number in the file */
58     enum mparset  inttype; /* which parser to use */
59     struct man    *pman; /* persistent man parser */
60     struct mdoc   *pmdoc; /* persistent mdoc parser */
61     struct man    *man; /* man parser */
62     struct mdoc   *mdoc; /* mdoc parser */
63     struct roff   *roff; /* roff parser (!NULL) */
64     int          reparse_count; /* finite interp. stack */
65     mmsg; /* warning/error message handler */
66     void         *arg; /* argument to mmsg */
67     const char   *file;
68     struct buf    *secondary;
69 };
70
71 static void      resize_buf(struct buf *, size_t);
72 static void      mparse_buf_r(struct mparse *, struct buf, int);
73 static void      mparse_readfd_r(struct mparse *, int, const char *, int);
74 static void      pset(const char *, int, struct mparse *);
75 static int       read_whole_file(const char *, int, struct buf *, int *);
76 static void      mparse_end(struct mparse *);
77
78 static const enum mandocerr  mandoclimits[MANDOCLEVEL_MAX] = {
79     MANDOCERR_OK,
80     MANDOCERR_WARNING,
81     MANDOCERR_WARNING,
82     MANDOCERR_ERROR,
83     MANDOCERR_FATAL,
84     MANDOCERR_MAX,
85     MANDOCERR_MAX
86 };
87
88 static const char * const   mandocerrs[MANDOCERR_MAX] = {
89     "ok",
90
91     "generic warning",
92
93     /* related to the prologue */
94     "no title in document",
95     "document title should be all caps",
96     "unknown manual section",
97     "date missing, using today's date",
98     "cannot parse date, using it verbatim",
99     "prologue macros out of order",
100    "duplicate prologue macro",
101    "macro not allowed in prologue",
102    "macro not allowed in body",
103
104    /* related to document structure */
105    ".so is fragile, better use ln(1)",
106    "NAME section must come first",
107    "bad NAME section contents",
108    "manual name not yet set",
109    "sections out of conventional order",
110    "duplicate section name",
111    "section not in conventional manual section",
112
113    /* related to macros and nesting */
114    "skipping obsolete macro",
115    "skipping paragraph macro",
116    "skipping no-space macro",
117    "blocks badly nested",
118    "child violates parent syntax",
119    "nested displays are not portable",
120    "already in literal mode",
121    "line scope broken",

```

```

123 /* related to missing macro arguments */
124 "skipping empty macro",
125 "argument count wrong",
126 "missing display type",
127 "list type must come first",
128 "tag lists require a width argument",
129 "missing font type",
130 "skipping end of block that is not open",

132 /* related to bad macro arguments */
133 "skipping argument",
134 "duplicate argument",
135 "duplicate display type",
136 "duplicate list type",
137 "unknown AT&T UNIX version",
138 "bad Boolean value",
139 "unknown font",
140 "unknown standard specifier",
141 "bad width argument",

143 /* related to plain text */
144 "blank line in non-literal context",
145 "tab in non-literal context",
146 "end of line whitespace",
147 "bad comment style",
148 "bad escape sequence",
149 "unterminated quoted string",

151 /* related to equations */
152 "unexpected literal in equation",
153
154 "generic error",

156 /* related to equations */
157 "unexpected equation scope closure",
158 "equation scope open on exit",
159 "overlapping equation scopes",
160 "unexpected end of equation",
161 "equation syntax error",

163 /* related to tables */
164 "bad table syntax",
165 "bad table option",
166 "bad table layout",
167 "no table layout cells specified",
168 "no table data cells specified",
169 "ignore data in cell",
170 "data block still open",
171 "ignoring extra data cells",

173 "input stack limit exceeded, infinite loop?",
174 "skipping bad character",
175 "escaped character not allowed in a name",
176 "skipping text before the first section header",
177 "skipping unknown macro",
178 "NOT IMPLEMENTED, please use groff: skipping request",
179 "argument count wrong",
180 "skipping end of block that is not open",
181 "missing end of block",
182 "scope open on exit",
183 "uname(3) system call failed",
184 "macro requires line argument(s)",
185 "macro requires body argument(s)",
186 "macro requires argument(s)",
187 "missing list type",
188 "line argument(s) will be lost",

```

```

189 "body argument(s) will be lost",

191 "generic fatal error",

193 "not a manual",
194 "column syntax is inconsistent",
195 "NOT IMPLEMENTED: .Bd -file",
196 "argument count wrong, violates syntax",
197 "child violates parent syntax",
198 "argument count wrong, violates syntax",
199 "NOT IMPLEMENTED: .so with absolute path or \"..\",
200 "no document body",
201 "no document prologue",
202 "static buffer exhausted",
203 };

205 static const char * const mandoclevels[MANDOCLEVEL_MAX] = {
206 "SUCCESS",
207 "RESERVED",
208 "WARNING",
209 "ERROR",
210 "FATAL",
211 "BADARG",
212 "SYSERR"
213 };

215 static void
216 resize_buf(struct buf *buf, size_t initial)
217 {

219     buf->sz = buf->sz > initial/2 ? 2 * buf->sz : initial;
220     buf->buf = mandoc_realloc(buf->buf, buf->sz);
221 }

223 static void
224 pset(const char *buf, int pos, struct mparse *curp)
225 {
226     int i;

228 /*
229  * Try to intuit which kind of manual parser should be used. If
230  * passed in by command-line (-man, -mdoc), then use that
231  * explicitly. If passed as -mandoc, then try to guess from the
232  * line: either skip dot-lines, use -mdoc when finding '.Dt', or
233  * default to -man, which is more lenient.
234  *
235  * Separate out pmdoc/pman from mdoc/man: the first persists
236  * through all parsers, while the latter is used per-parse.
237  */

239     if ('.' == buf[0] || '\'' == buf[0]) {
240         for (i = 1; buf[i]; i++)
241             if (' ' != buf[i] && '\t' != buf[i])
242                 break;
243             if ('\0' == buf[i])
244                 return;
245     }

247     switch (curp->inttype) {
248     case (MPARSE_MDOC):
249         if (NULL == curp->pmdoc)
250             curp->pmdoc = mdoc_alloc(curp->roff, curp);
251         assert(curp->pmdoc);
252         curp->mdoc = curp->pmdoc;
253         return;
254     case (MPARSE_MAN):

```



```

255     if (NULL == curp->pman)
256         curp->pman = man_alloc(curp->roff, curp);
257     assert(curp->pman);
258     curp->man = curp->pman;
259     return;
260 default:
261     break;
262 }

264 if (pos >= 3 && 0 == memcmp(buf, ".Dd", 3)) {
265     if (NULL == curp->pmdoc)
266         curp->pmdoc = mdoc_alloc(curp->roff, curp);
267     assert(curp->pmdoc);
268     curp->mdoc = curp->pmdoc;
269     return;
270 }

272 if (NULL == curp->pman)
273     curp->pman = man_alloc(curp->roff, curp);
274 assert(curp->pman);
275 curp->man = curp->pman;
276 }

278 /*
279  * Main parse routine for an opened file. This is called for each
280  * opened file and simply loops around the full input file, possibly
281  * nesting (i.e., with 'so').
282  */
283 static void
284 mparse_buf_r(struct mparse *curp, struct buf blk, int start)
285 {
286     const struct tbl_span *span;
287     struct buf ln;
288     enum rofferr rr;
289     int i, of, rc;
290     int pos; /* byte number in the ln buffer */
291     int lnn; /* line number in the real file */
292     unsigned char c;

294     memset(&ln, 0, sizeof(struct buf));

296     lnn = curp->line;
297     pos = 0;

299     for (i = 0; i < (int)blk.sz; ) {
300         if (0 == pos && '\0' == blk.buf[i])
301             break;

303         if (start) {
304             curp->line = lnn;
305             curp->reparse_count = 0;
306         }

308         while (i < (int)blk.sz && (start || '\0' != blk.buf[i])) {

310             /*
311              * When finding an unescaped newline character,
312              * leave the character loop to process the line.
313              * Skip a preceding carriage return, if any.
314              */

316             if ('\r' == blk.buf[i] && i + 1 < (int)blk.sz &&
317                 '\n' == blk.buf[i + 1])
318                 ++i;
319             if ('\n' == blk.buf[i]) {
320                 ++i;

```

```

321                 ++lnn;
322                 break;
323             }

325         /*
326          * Warn about bogus characters. If you're using
327          * non-ASCII encoding, you're screwing your
328          * readers. Since I'd rather this not happen,
329          * I'll be helpful and replace these characters
330          * with "?", so we don't display gibberish.
331          * Note to manual writers: use special characters.
332          */

334         c = (unsigned char) blk.buf[i];

336         if ( ! (isascii(c) &&
337             (isgraph(c) || isblank(c)))) {
338             mandoc_msg(MANDOCERR_BADCHAR, curp,
339                 curp->line, pos, NULL);
340             i++;
341             if (pos >= (int)ln.sz)
342                 resize_buf(&ln, 256);
343             ln.buf[pos++] = '?';
344             continue;
345         }

347         /* Trailing backslash = a plain char. */

349         if ('\\" != blk.buf[i] || i + 1 == (int)blk.sz) {
350             if (pos >= (int)ln.sz)
351                 resize_buf(&ln, 256);
352             ln.buf[pos++] = blk.buf[i++];
353             continue;
354         }

356         /*
357          * Found escape and at least one other character.
358          * When it's a newline character, skip it.
359          * When there is a carriage return in between,
360          * skip that one as well.
361          */

363         if ('\r' == blk.buf[i + 1] && i + 2 < (int)blk.sz &&
364             '\n' == blk.buf[i + 2])
365             ++i;
366         if ('\n' == blk.buf[i + 1]) {
367             i += 2;
368             ++lnn;
369             continue;
370         }

372         if ('"' == blk.buf[i + 1] || '#' == blk.buf[i + 1]) {
373             i += 2;
374             /* Comment, skip to end of line */
375             for (; i < (int)blk.sz; ++i) {
376                 if ('\n' == blk.buf[i]) {
377                     ++i;
378                     ++lnn;
379                     break;
380                 }
381             }

383             /* Backout trailing whitespaces */
384             for (; pos > 0; --pos) {
385                 if (ln.buf[pos - 1] != ' ')
386                     break;

```

```

387         if (pos > 2 && ln.buf[pos - 2] == '\\\\')
388             break;
389     }
390     break;
391 }
392
393     /* Some other escape sequence, copy & cont. */
394
395     if (pos + 1 >= (int)ln.sz)
396         resize_buf(&ln, 256);
397
398     ln.buf[pos++] = blk.buf[i++];
399     ln.buf[pos++] = blk.buf[i++];
400 }
401
402 if (pos >= (int)ln.sz)
403     resize_buf(&ln, 256);
404
405 ln.buf[pos] = '\\0';
406
407 /*
408  * A significant amount of complexity is contained by
409  * the roff preprocessor. It's line-oriented but can be
410  * expressed on one line, so we need at times to
411  * readjust our starting point and re-run it. The roff
412  * preprocessor can also readjust the buffers with new
413  * data, so we pass them in wholesale.
414  */
415
416 of = 0;
417
418 /*
419  * Maintain a lookaside buffer of all parsed lines. We
420  * only do this if mparse_keep() has been invoked (the
421  * buffer may be accessed with mparse_getkeep()).
422  */
423
424 if (curp->secondary) {
425     curp->secondary->buf =
426         mandoc_realloc
427             (curp->secondary->buf,
428             curp->secondary->sz + pos + 2);
429     memcpy(curp->secondary->buf +
430           curp->secondary->sz,
431           ln.buf, pos);
432     curp->secondary->sz += pos;
433     curp->secondary->buf
434         [curp->secondary->sz] = '\\n';
435     curp->secondary->sz++;
436     curp->secondary->buf
437         [curp->secondary->sz] = '\\0';
438 }
439 rerun:
440 rr = roff_parseln
441     (curp->roff, curp->line,
442     &ln.buf, &ln.sz, of, &of);
443
444 switch (rr) {
445 case (ROFF_REPARSE):
446     if (REPARSE_LIMIT >= ++curp->reparse_count)
447         mparse_buf_r(curp, ln, 0);
448     else
449         mandoc_msg(MANDOCERR_ROFFLOOP, curp,
450                 curp->line, pos, NULL);
451     pos = 0;
452     continue;

```

```

453     case (ROFF_APPEND):
454         pos = (int)strlen(ln.buf);
455         continue;
456     case (ROFF_RERUN):
457         goto rerun;
458     case (ROFF_IGN):
459         pos = 0;
460         continue;
461     case (ROFF_ERR):
462         assert(MANDOCLEVEL_FATAL <= curp->file_status);
463         break;
464     case (ROFF_SO):
465         /*
466          * We remove 'so' clauses from our lookaside
467          * buffer because we're going to descend into
468          * the file recursively.
469          */
470         if (curp->secondary)
471             curp->secondary->sz -= pos + 1;
472         mparse_readfd_r(curp, -1, ln.buf + of, 1);
473         if (MANDOCLEVEL_FATAL <= curp->file_status)
474             break;
475         pos = 0;
476         continue;
477     default:
478         break;
479 }
480
481 /*
482  * If we encounter errors in the recursive parse, make
483  * sure we don't continue parsing.
484  */
485
486 if (MANDOCLEVEL_FATAL <= curp->file_status)
487     break;
488
489 /*
490  * If input parsers have not been allocated, do so now.
491  * We keep these instanced between parsers, but set them
492  * locally per parse routine since we can use different
493  * parsers with each one.
494  */
495
496 if (! (curp->man || curp->mdoc))
497     pset(ln.buf + of, pos - of, curp);
498
499 /*
500  * Lastly, push down into the parsers themselves. One
501  * of these will have already been set in the pset()
502  * routine.
503  * If libroff returns ROFF_TBL, then add it to the
504  * currently open parse. Since we only get here if
505  * there does exist data (see tbl_data.c), we're
506  * guaranteed that something's been allocated.
507  * Do the same for ROFF_EQN.
508  */
509
510 rc = -1;
511
512 if (ROFF_TBL == rr)
513     while (NULL != (span = roff_span(curp->roff))) {
514         rc = curp->man ?
515             man_addspan(curp->man, span) :
516             mdoc_addspan(curp->mdoc, span);
517         if (0 == rc)
518             break;

```

```

519     }
520     else if (ROFF_EQN == rr)
521         rc = curp->mdoc ?
522             mdoc_addeqn(curp->mdoc,
523                 roff_eqn(curp->roff)) :
524             man_addeqn(curp->man,
525                 roff_eqn(curp->roff));
526     else if (curp->man || curp->mdoc)
527         rc = curp->man ?
528             man_parseln(curp->man,
529                 curp->line, ln.buf, of) :
530             mdoc_parseln(curp->mdoc,
531                 curp->line, ln.buf, of);
532
533     if (0 == rc) {
534         assert(MANDOCLEVEL_FATAL <= curp->file_status);
535         break;
536     }
537
538     /* Temporary buffers typically are not full. */
539
540     if (0 == start && '\0' == blk.buf[i])
541         break;
542
543     /* Start the next input line. */
544
545     pos = 0;
546 }
547
548 free(ln.buf);
549 }
550
551 static int
552 read_whole_file(const char *file, int fd, struct buf *fb, int *with_mmap)
553 {
554     size_t    off;
555     ssize_t   ssz;
556
557 #ifdef HAVE_MMAP
558     struct stat st;
559     if (-1 == fstat(fd, &st)) {
560         perror(file);
561         return(0);
562     }
563
564     /*
565      * If we're a regular file, try just reading in the whole entry
566      * via mmap(). This is faster than reading it into blocks, and
567      * since each file is only a few bytes to begin with, I'm not
568      * concerned that this is going to tank any machines.
569      */
570
571     if (S_ISREG(st.st_mode)) {
572         if (st.st_size >= (1U << 31)) {
573             fprintf(stderr, "%s: input too large\n", file);
574             return(0);
575         }
576         *with_mmap = 1;
577         fb->sz = (size_t)st.st_size;
578         fb->buf = mmap(NULL, fb->sz, PROT_READ,
579             MAP_FILE|MAP_SHARED, fd, 0);
580         if (fb->buf != MAP_FAILED)
581             return(1);
582     }
583 #endif

```

```

585     /*
586      * If this isn't a regular file (like, say, stdin), then we must
587      * go the old way and just read things in bit by bit.
588      */
589
590     *with_mmap = 0;
591     off = 0;
592     fb->sz = 0;
593     fb->buf = NULL;
594     for (;;) {
595         if (off == fb->sz) {
596             if (fb->sz == (1U << 31)) {
597                 fprintf(stderr, "%s: input too large\n", file);
598                 break;
599             }
600             resize_buf(fb, 65536);
601         }
602         ssz = read(fd, fb->buf + (int)off, fb->sz - off);
603         if (ssz == 0) {
604             fb->sz = off;
605             return(1);
606         }
607         if (ssz == -1) {
608             perror(file);
609             break;
610         }
611         off += (size_t)ssz;
612     }
613
614     free(fb->buf);
615     fb->buf = NULL;
616     return(0);
617 }
618
619 static void
620 mparse_end(struct mparse *curp)
621 {
622
623     if (MANDOCLEVEL_FATAL <= curp->file_status)
624         return;
625
626     if (curp->mdoc && ! mdoc_endparse(curp->mdoc)) {
627         assert(MANDOCLEVEL_FATAL <= curp->file_status);
628         return;
629     }
630
631     if (curp->man && ! man_endparse(curp->man)) {
632         assert(MANDOCLEVEL_FATAL <= curp->file_status);
633         return;
634     }
635
636     if (! (curp->man || curp->mdoc)) {
637         mandoc_msg(MANDOCERR_NOTMANUAL, curp, 1, 0, NULL);
638         curp->file_status = MANDOCLEVEL_FATAL;
639         return;
640     }
641
642     roff_endparse(curp->roff);
643 }
644
645 static void
646 mparse_parse_buffer(struct mparse *curp, struct buf blk, const char *file,
647     int re)
648 {
649     const char    *svfile;

```

```

651      /* Line number is per-file. */
652      svfile = curp->file;
653      curp->file = file;
654      curp->line = 1;

656      mparse_buf_r(curp, blk, 1);

658      if (0 == re && MANDOCLEVEL_FATAL > curp->file_status)
659          mparse_end(curp);

661      curp->file = svfile;
662  }

664  enum mandoclevel
665  mparse_readmem(struct mparse *curp, const void *buf, size_t len,
666                const char *file)
667  {
668      struct buf blk;

670      blk.buf = UNCONST(buf);
671      blk.sz = len;

673      mparse_parse_buffer(curp, blk, file, 0);
674      return(curp->file_status);
675  }

677  static void
678  mparse_readfd_r(struct mparse *curp, int fd, const char *file, int re)
679  {
680      struct buf      blk;
681      int              with_mmap;

683      if (-1 == fd)
684          if (-1 == (fd = open(file, O_RDONLY, 0))) {
685              perror(file);
686              curp->file_status = MANDOCLEVEL_SYSERR;
687              return;
688          }
689      /*
690       * Run for each opened file; may be called more than once for
691       * each full parse sequence if the opened file is nested (i.e.,
692       * from 'so'). Simply sucks in the whole file and moves into
693       * the parse phase for the file.
694       */

696      if (! read_whole_file(file, fd, &blk, &with_mmap)) {
697          curp->file_status = MANDOCLEVEL_SYSERR;
698          return;
699      }

701      mparse_parse_buffer(curp, blk, file, re);

703  #ifdef HAVE_MMAP
704      if (with_mmap)
705          munmap(blk.buf, blk.sz);
706      else
707  #endif
708          free(blk.buf);

710      if (STDIN_FILENO != fd && -1 == close(fd))
711          perror(file);
712  }

714  enum mandoclevel
715  mparse_readfd(struct mparse *curp, int fd, const char *file)
716  {

```

```

718      mparse_readfd_r(curp, fd, file, 0);
719      return(curp->file_status);
720  }

722  struct mparse *
723  mparse_alloc(enum mparset inttype, enum mandoclevel wlevel, mandocmsg mmsg, void
724  {
725      struct mparse *curp;

727      assert(wlevel <= MANDOCLEVEL_FATAL);

729      curp = mandoc_calloc(1, sizeof(struct mparse));

731      curp->wlevel = wlevel;
732      curp->mmsg = mmsg;
733      curp->arg = arg;
734      curp->inttype = inttype;

736      curp->roff = roff_alloc(curp);
737      return(curp);
738  }

740  void
741  mparse_reset(struct mparse *curp)
742  {
744      roff_reset(curp->roff);

746      if (curp->mdoc)
747          mdoc_reset(curp->mdoc);
748      if (curp->man)
749          man_reset(curp->man);
750      if (curp->secondary)
751          curp->secondary->sz = 0;

753      curp->file_status = MANDOCLEVEL_OK;
754      curp->mdoc = NULL;
755      curp->man = NULL;
756  }

758  void
759  mparse_free(struct mparse *curp)
760  {
762      if (curp->pmdoc)
763          mdoc_free(curp->pmdoc);
764      if (curp->pman)
765          man_free(curp->pman);
766      if (curp->roff)
767          roff_free(curp->roff);
768      if (curp->secondary)
769          free(curp->secondary->buf);

771      free(curp->secondary);
772      free(curp);
773  }

775  void
776  mparse_result(struct mparse *curp, struct mdoc **mdoc, struct man **man)
777  {
779      if (mdoc)
780          *mdoc = curp->mdoc;
781      if (man)
782          *man = curp->man;

```

```
783 }

785 void
786 mandoc_vmsg(enum mandocerr t, struct mparse *m,
787             int ln, int pos, const char *fmt, ...)
788 {
789     char          buf[256];
790     va_list       ap;

792     va_start(ap, fmt);
793     vsnprintf(buf, sizeof(buf) - 1, fmt, ap);
794     va_end(ap);

796     mandoc_msg(t, m, ln, pos, buf);
797 }

799 void
800 mandoc_msg(enum mandocerr er, struct mparse *m,
801            int ln, int col, const char *msg)
802 {
803     enum mandoclevel level;

805     level = MANDOCLEVEL_FATAL;
806     while (er < mandoclimits[level])
807         level--;

809     if (level < m->wlevel)
810         return;

812     if (m->mmsg)
813         (*m->mmsg)(er, level, m->file, ln, col, msg);

815     if (m->file_status < level)
816         m->file_status = level;
817 }

819 const char *
820 mparse_strerror(enum mandocerr er)
821 {
823     return(mandocerrs[er]);
824 }

826 const char *
827 mparse_strlevel(enum mandoclevel lvl)
828 {
829     return(mandoclevels[lvl]);
830 }

832 void
833 mparse_keep(struct mparse *p)
834 {
836     assert(NULL == p->secondary);
837     p->secondary = mandoc_calloc(1, sizeof(struct buf));
838 }

840 const char *
841 mparse_getkeep(const struct mparse *p)
842 {
844     assert(p->secondary);
845     return(p->secondary->sz ? p->secondary->buf : NULL);
846 }
```

```

*****
37963 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/roff.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: roff.c,v 1.172 2011/10/24 21:41:45 schwarze Exp $ */
2 /*
3  * Copyright (c) 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2010, 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHORS DISCLAIM ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <assert.h>
23 #include <ctype.h>
24 #include <stdlib.h>
25 #include <string.h>
26
27 #include "mandoc.h"
28 #include "libroff.h"
29 #include "libmandoc.h"
30
31 /* Maximum number of nested if-else conditionals. */
32 #define RSTACK_MAX 128
33
34 /* Maximum number of string expansions per line, to break infinite loops. */
35 #define EXPAND_LIMIT 1000
36
37 enum rofft {
38     ROFF_ad,
39     ROFF_am,
40     ROFF_ami,
41     ROFF_aml,
42     ROFF_de,
43     ROFF_dei,
44     ROFF_del,
45     ROFF_ds,
46     ROFF_el,
47     ROFF_hy,
48     ROFF_ie,
49     ROFF_if,
50     ROFF_ig,
51     ROFF_it,
52     ROFF_ne,
53     ROFF_nh,
54     ROFF_nr,
55     ROFF_ns,
56     ROFF_ps,

```

```

57     ROFF_rm,
58     ROFF_so,
59     ROFF_ta,
60     ROFF_tr,
61     ROFF_TS,
62     ROFF_TE,
63     ROFF_T_,
64     ROFF_EQ,
65     ROFF_EN,
66     ROFF_cblock,
67     ROFF_ccond,
68     ROFF_USERDEF,
69     ROFF_MAX
70 };
71
72 enum roffrule {
73     ROFFRULE_ALLOW,
74     ROFFRULE_DENY
75 };
76
77 /*
78  * A single register entity. If "set" is zero, the value of the
79  * register should be the default one, which is per-register.
80  * Registers are assumed to be unsigned ints for now.
81  */
82 struct reg {
83     int set; /* whether set or not */
84     unsigned int u; /* unsigned integer */
85 };
86
87 /*
88  * An incredibly-simple string buffer.
89  */
90 struct roffstr {
91     char *p; /* nil-terminated buffer */
92     size_t sz; /* saved strlen(p) */
93 };
94
95 /*
96  * A key-value roffstr pair as part of a singly-linked list.
97  */
98 struct roffkv {
99     struct roffstr key;
100    struct roffstr val;
101    struct roffkv *next; /* next in list */
102 };
103
104 struct roff {
105     struct mparse *parse; /* parse point */
106     struct roffnode *last; /* leaf of stack */
107     enum roffrule rstack[RSTACK_MAX]; /* stack of 'ie' rules */
108     int rstackpos; /* position in rstack */
109     struct reg regs[REG_MAX];
110     struct roffkv *strtab; /* user-defined strings & macros */
111     struct roffkv *xmbtab; /* multi-byte trans table ('tr') */
112     struct roffstr *xtab; /* single-byte trans table ('tr') */
113     const char *current_string; /* value of last called user macro */
114     struct tbl_node *first_tbl; /* first table parsed */
115     struct tbl_node *last_tbl; /* last table parsed */
116     struct tbl_node *tbl; /* current table being parsed */
117     struct eqn_node *last_eqn; /* last equation parsed */
118     struct eqn_node *first_eqn; /* first equation parsed */
119     struct eqn_node *eqn; /* current equation being parsed */
120 };
121
122 struct roffnode {

```

```

123     enum roffft      tok; /* type of node */
124     struct roffnode *parent; /* up one in stack */
125     int              line; /* parse line */
126     int              col; /* parse col */
127     char             *name; /* node name, e.g. macro name */
128     char             *end; /* end-rules: custom token */
129     int              endspar; /* end-rules: next-line or infity */
130     enum roffrule    rule; /* current evaluation rule */
131 };

133 #define ROFF_ARGS      struct roff *r, /* parse ctx */ \
134                       enum roffft tok, /* tok of macro */ \
135                       char **bufp, /* input buffer */ \
136                       size_t *szp, /* size of input buffer */ \
137                       int ln, /* parse line */ \
138                       int ppos, /* original pos in buffer */ \
139                       int pos, /* current pos in buffer */ \
140                       int *offs /* reset offset of buffer data */

142 typedef enum rofferr (*roffproc)(ROFF_ARGS);

144 struct roffmac {
145     const char      *name; /* macro name */
146     roffproc        proc; /* process new macro */
147     roffproc        text; /* process as child text of macro */
148     roffproc        sub; /* process as child of macro */
149     int             flags;
150 #define ROFFMAC_STRUCT (1 << 0) /* always interpret */
151     struct roffmac *next;
152 };

154 struct predef {
155     const char      *name; /* predefined input name */
156     const char      *str; /* replacement symbol */
157 };

159 #define PREDEF(__name, __str) \
160     { (__name), (__str) },

162 static enum roffft      roffhash_find(const char *, size_t);
163 static void             roffhash_init(void);
164 static void             roffnode_cleanscope(struct roff *);
165 static void             roffnode_pop(struct roff *);
166 static void             roffnode_push(struct roff *, enum roffft,
167                                     const char *, int, int);
168 static enum rofferr     roff_block(ROFF_ARGS);
169 static enum rofferr     roff_block_text(ROFF_ARGS);
170 static enum rofferr     roff_block_sub(ROFF_ARGS);
171 static enum rofferr     roff_cblock(ROFF_ARGS);
172 static enum rofferr     roff_ccond(ROFF_ARGS);
173 static enum rofferr     roff_cond(ROFF_ARGS);
174 static enum rofferr     roff_cond_text(ROFF_ARGS);
175 static enum rofferr     roff_cond_sub(ROFF_ARGS);
176 static enum rofferr     roff_ds(ROFF_ARGS);
177 static enum roffrule    roff_evalcond(const char *, int *);
178 static void             roff_freel(struct roff *);
179 static void             roff_freestr(struct roffkv *);
180 static char             *roff_getname(struct roff *, char **, int, int);
181 static const char      *roff_getstrn(const struct roff *,
182                                     const char *, size_t);
183 static enum rofferr     roff_line_ignore(ROFF_ARGS);
184 static enum rofferr     roff_nr(ROFF_ARGS);
185 static void             roff_openeqn(struct roff *, const char *,
186                                     int, int, const char *);
187 static enum roffft      roff_parse(struct roff *, const char *, int *);
188 static enum rofferr     roff_parsertext(char *);

```

```

189 static enum rofferr     roff_res(struct roff *,
190                                 char **, size_t *, int, int);
191 static enum rofferr     roff_rm(ROFF_ARGS);
192 static void             roff_setstr(struct roff *,
193                                 const char *, const char *, int);
194 static void             roff_setstrn(struct roffkv **, const char *,
195                                    size_t, const char *, size_t, int);
196 static enum rofferr     roff_so(ROFF_ARGS);
197 static enum rofferr     roff_tr(ROFF_ARGS);
198 static enum rofferr     roff_TE(ROFF_ARGS);
199 static enum rofferr     roff_TS(ROFF_ARGS);
200 static enum rofferr     roff_EQ(ROFF_ARGS);
201 static enum rofferr     roff_EN(ROFF_ARGS);
202 static enum rofferr     roff_T_(ROFF_ARGS);
203 static enum rofferr     roff_userdef(ROFF_ARGS);

205 /* See roffhash_find() */

207 #define ASCII_HI      126
208 #define ASCII_LO      33
209 #define HASHWIDTH    (ASCII_HI - ASCII_LO + 1)

211 static struct roffmac *hash[HASHWIDTH];

213 static struct roffmac roffs[ROFF_MAX] = {
214     {"ad", roff_line_ignore, NULL, NULL, 0, NULL },
215     {"am", roff_block, roff_block_text, roff_block_sub, 0, NULL },
216     {"am1", roff_block, roff_block_text, roff_block_sub, 0, NULL },
217     {"aml", roff_block, roff_block_text, roff_block_sub, 0, NULL },
218     {"de", roff_block, roff_block_text, roff_block_sub, 0, NULL },
219     {"dei", roff_block, roff_block_text, roff_block_sub, 0, NULL },
220     {"del", roff_block, roff_block_text, roff_block_sub, 0, NULL },
221     {"ds", roff_ds, NULL, NULL, 0, NULL },
222     {"el", roff_cond, roff_cond_text, roff_cond_sub, ROFFMAC_STRUCT, NULL },
223     {"hy", roff_line_ignore, NULL, NULL, 0, NULL },
224     {"ie", roff_cond, roff_cond_text, roff_cond_sub, ROFFMAC_STRUCT, NULL },
225     {"if", roff_cond, roff_cond_text, roff_cond_sub, ROFFMAC_STRUCT, NULL },
226     {"ig", roff_block, roff_block_text, roff_block_sub, 0, NULL },
227     {"it", roff_line_ignore, NULL, NULL, 0, NULL },
228     {"ne", roff_line_ignore, NULL, NULL, 0, NULL },
229     {"nh", roff_line_ignore, NULL, NULL, 0, NULL },
230     {"nr", roff_nr, NULL, NULL, 0, NULL },
231     {"ns", roff_line_ignore, NULL, NULL, 0, NULL },
232     {"ps", roff_line_ignore, NULL, NULL, 0, NULL },
233     {"rm", roff_rm, NULL, NULL, 0, NULL },
234     {"so", roff_so, NULL, NULL, 0, NULL },
235     {"ta", roff_line_ignore, NULL, NULL, 0, NULL },
236     {"tr", roff_tr, NULL, NULL, 0, NULL },
237     {"TS", roff_TS, NULL, NULL, 0, NULL },
238     {"TE", roff_TE, NULL, NULL, 0, NULL },
239     {"T&", roff_T_, NULL, NULL, 0, NULL },
240     {"EQ", roff_EQ, NULL, NULL, 0, NULL },
241     {"EN", roff_EN, NULL, NULL, 0, NULL },
242     {".", roff_cblock, NULL, NULL, 0, NULL },
243     {"\\", roff_ccond, NULL, NULL, 0, NULL },
244     {NULL, roff_userdef, NULL, NULL, 0, NULL },
245 };

247 /* Array of injected predefined strings. */
248 #define PREDEFS_MAX    38
249 static const struct predef predefs[PREDEFS_MAX] = {
250     #include "predefs.in"
251 };

253 /* See roffhash_find() */
254 #define ROFF_HASH(p)    (p[0] - ASCII_LO)

```

```

256 static void
257 roffhash_init(void)
258 {
259     struct roffmac *n;
260     int          buc, i;

262     for (i = 0; i < (int)ROFF_USERDEF; i++) {
263         assert(roffs[i].name[0] >= ASCII_LO);
264         assert(roffs[i].name[0] <= ASCII_HI);

266         buc = ROFF_HASH(roffs[i].name);

268         if (NULL != (n = hash[buc])) {
269             for ( ; n->next; n = n->next)
270                 /* Do nothing. */ ;
271             n->next = &roffs[i];
272         } else
273             hash[buc] = &roffs[i];
274     }
275 }

277 /*
278  * Look up a roff token by its name. Returns ROFF_MAX if no macro by
279  * the nil-terminated string name could be found.
280  */
281 static enum roffft
282 roffhash_find(const char *p, size_t s)
283 {
284     int          buc;
285     struct roffmac *n;

287     /*
288      * libroff has an extremely simple hashtable, for the time
289      * being, which simply keys on the first character, which must
290      * be printable, then walks a chain. It works well enough until
291      * optimised.
292      */

294     if (p[0] < ASCII_LO || p[0] > ASCII_HI)
295         return(ROFF_MAX);

297     buc = ROFF_HASH(p);

299     if (NULL == (n = hash[buc]))
300         return(ROFF_MAX);
301     for ( ; n; n = n->next)
302         if (0 == strncmp(n->name, p, s) && '\0' == n->name[(int)s])
303             return((enum roffft)(n - roffs));

305     return(ROFF_MAX);
306 }

309 /*
310  * Pop the current node off of the stack of roff instructions currently
311  * pending.
312  */
313 static void
314 roffnode_pop(struct roff *r)
315 {
316     struct roffnode *p;

318     assert(r->last);
319     p = r->last;

```

```

321     r->last = r->last->parent;
322     free(p->name);
323     free(p->end);
324     free(p);
325 }

328 /*
329  * Push a roff node onto the instruction stack. This must later be
330  * removed with roffnode_pop().
331  */
332 static void
333 roffnode_push(struct roff *r, enum roffft tok, const char *name,
334              int line, int col)
335 {
336     struct roffnode *p;

338     p = mandoc_calloc(1, sizeof(struct roffnode));
339     p->tok = tok;
340     if (name)
341         p->name = mandoc_strdup(name);
342     p->parent = r->last;
343     p->line = line;
344     p->col = col;
345     p->rule = p->parent ? p->parent->rule : ROFFRULE_DENY;

347     r->last = p;
348 }

351 static void
352 roff_freel(struct roff *r)
353 {
354     struct tbl_node *t;
355     struct eqn_node *e;
356     int          i;

358     while (NULL != (t = r->first_tbl)) {
359         r->first_tbl = t->next;
360         tbl_free(t);
361     }

363     r->first_tbl = r->last_tbl = r->tbl = NULL;

365     while (NULL != (e = r->first_eqn)) {
366         r->first_eqn = e->next;
367         eqn_free(e);
368     }

370     r->first_eqn = r->last_eqn = r->eqn = NULL;

372     while (r->last)
373         roffnode_pop(r);

375     roff_freestr(r->strtab);
376     roff_freestr(r->xmbtab);

378     r->strtab = r->xmbtab = NULL;

380     if (r->xtab)
381         for (i = 0; i < 128; i++)
382             free(r->xtab[i].p);

384     free(r->xtab);
385     r->xtab = NULL;
386 }

```



```

388 void
389 roff_reset(struct roff *r)
390 {
391     int            i;
392
393     roff_freel(r);
394
395     memset(&r->regs, 0, sizeof(struct reg) * REG_MAX);
396
397     for (i = 0; i < PREDEFS_MAX; i++)
398         roff_setstr(r, predefs[i].name, predefs[i].str, 0);
399 }
400
401 void
402 roff_free(struct roff *r)
403 {
404     roff_freel(r);
405     free(r);
406 }
407
408 }
409
410 struct roff *
411 roff_alloc(struct mparse *parse)
412 {
413     struct roff    *r;
414     int            i;
415
416     r = mandoc_calloc(1, sizeof(struct roff));
417     r->parse = parse;
418     r->rstackpos = -1;
419
420     roffhash_init();
421
422     for (i = 0; i < PREDEFS_MAX; i++)
423         roff_setstr(r, predefs[i].name, predefs[i].str, 0);
424
425     return(r);
426 }
427
428 /*
429 * Pre-filter each and every line for reserved words (one beginning with
430 * '\*', e.g., '\*(ab)'). These must be handled before the actual line
431 * is processed.
432 * This also checks the syntax of regular escapes.
433 */
434 static enum rofferr
435 roff_res(struct roff *r, char **bufp, size_t *szp, int ln, int pos)
436 {
437     enum mandoc_esc  esc;
438     const char       *stesc; /* start of an escape sequence ('\') */
439     const char       *stnam; /* start of the name, after "[(** */
440     const char       *cp;    /* end of the name, e.g. before ']' */
441     const char       *res;   /* the string to be substituted */
442     int              i, maxl, expand_count;
443     size_t           nsz;
444     char             *n;
445
446     expand_count = 0;
447
448 again:
449     cp = *bufp + pos;
450     while (NULL != (cp = strchr(cp, '\\'))) {
451         stesc = cp++;

```

```

452         /*
453          * The second character must be an asterisk.
454          * If it isn't, skip it anyway: It is escaped,
455          * so it can't start another escape sequence.
456          */
457         if ('\0' == *cp)
458             return(ROFF_CONT);
459
460         if ('*' != *cp) {
461             res = cp;
462             esc = mandoc_escape(&cp, NULL, NULL);
463             if (ESCAPE_ERROR != esc)
464                 continue;
465             cp = res;
466             mandoc_msg
467                 (MANDOCERR_BADESCAPE, r->parse,
468                  ln, (int)(stesc - *bufp), NULL);
469             return(ROFF_CONT);
470         }
471
472         cp++;
473
474         /*
475          * The third character decides the length
476          * of the name of the string.
477          * Save a pointer to the name.
478          */
479         switch (*cp) {
480             case ('\0'):
481                 return(ROFF_CONT);
482             case ('('):
483                 cp++;
484                 maxl = 2;
485                 break;
486             case ('['):
487                 cp++;
488                 maxl = 0;
489                 break;
490             default:
491                 maxl = 1;
492                 break;
493         }
494         stnam = cp;
495
496         /* Advance to the end of the name. */
497         for (i = 0; 0 == maxl || i < maxl; i++, cp++) {
498             if ('\0' == *cp) {
499                 mandoc_msg
500                     (MANDOCERR_BADESCAPE,
501                      r->parse, ln,
502                      (int)(stesc - *bufp), NULL);
503                 return(ROFF_CONT);
504             }
505             if (0 == maxl && '[' == *cp)
506                 break;
507         }
508
509         /*
510          * Retrieve the replacement string; if it is
511          * undefined, resume searching for escapes.
512          */

```

```

519     res = roff_getstrn(r, stnam, (size_t)1);
521     if (NULL == res) {
522         mandoc_msg
523             (MANDOCERR_BADESCAPE, r->parse,
524              ln, (int)(stesc - *bufp), NULL);
525         res = "";
526     }
528     /* Replace the escape sequence by the string. */
530     pos = stesc - *bufp;
532     nsz = *szp + strlen(res) + 1;
533     n = mandoc_malloc(nsz);
535     strncpy(n, *bufp, (size_t)(stesc - *bufp + 1));
536     strcat(n, res, nsz);
537     strcat(n, cp + (maxl ? 0 : 1), nsz);
539     free(*bufp);
541     *bufp = n;
542     *szp = nsz;
544     if (EXPAND_LIMIT >= ++expand_count)
545         goto again;
547     /* Just leave the string unexpanded. */
548     mandoc_msg(MANDOCERR_ROFFLOOP, r->parse, ln, pos, NULL);
549     return(ROFF_IGN);
550 }
551 return(ROFF_CONT);
552 }
554 /*
555  * Process text streams: convert all breakable hyphens into ASCII_HYPH.
556  */
557 static enum rofferr
558 roff_parsertext(char *p)
559 {
560     size_t      sz;
561     const char  *start;
562     enum mandoc_esc  esc;
564     start = p;
566     while ('\0' != *p) {
567         sz = strcspn(p, "-\\");
568         p += sz;
570         if ('\0' == *p)
571             break;
573         if ('\\' == *p) {
574             /* Skip over escapes. */
575             p++;
576             esc = mandoc_escape
577                 ((const char **)&, NULL, NULL);
578             if (ESCAPE_ERROR == esc)
579                 break;
580             continue;
581         } else if (p == start) {
582             p++;
583             continue;
584         }

```

```

586         if (isalpha((unsigned char)p[-1]) &&
587             isalpha((unsigned char)p[1]))
588             *p = ASCII_HYPH;
589         p++;
590     }
592     return(ROFF_CONT);
593 }
595 enum rofferr
596 roff_parseln(struct roff *r, int ln, char **bufp,
597             size_t *szp, int pos, int *offs)
598 {
599     enum roffft      t;
600     enum rofferr     e;
601     int              ppos, ctl;
603     /*
604      * Run the reserved-word filter only if we have some reserved
605      * words to fill in.
606      */
608     e = roff_res(r, bufp, szp, ln, pos);
609     if (ROFF_IGN == e)
610         return(e);
611     assert(ROFF_CONT == e);
613     ppos = pos;
614     ctl = mandoc_getcontrol(*bufp, &pos);
616     /*
617      * First, if a scope is open and we're not a macro, pass the
618      * text through the macro's filter. If a scope isn't open and
619      * we're not a macro, just let it through.
620      * Finally, if there's an equation scope open, divert it into it
621      * no matter our state.
622      */
624     if (r->last && !ctl) {
625         t = r->last->tok;
626         assert(roffs[t].text);
627         e = (*roffs[t].text)
628             (r, t, bufp, szp, ln, pos, pos, offs);
629         assert(ROFF_IGN == e || ROFF_CONT == e);
630         if (ROFF_CONT != e)
631             return(e);
632         if (r->eqn)
633             return(eqn_read(&r->eqn, ln, *bufp, pos, offs));
634         if (r->tbl)
635             return(tbl_read(r->tbl, ln, *bufp, pos));
636         return(roff_parsertext(*bufp + pos));
637     } else if (!ctl) {
638         if (r->eqn)
639             return(eqn_read(&r->eqn, ln, *bufp, pos, offs));
640         if (r->tbl)
641             return(tbl_read(r->tbl, ln, *bufp, pos));
642         return(roff_parsertext(*bufp + pos));
643     } else if (r->eqn)
644         return(eqn_read(&r->eqn, ln, *bufp, ppos, offs));
646     /*
647      * If a scope is open, go to the child handler for that macro,
648      * as it may want to preprocess before doing anything with it.
649      * Don't do so if an equation is open.
650      */

```

```

652     if (r->last) {
653         t = r->last->tok;
654         assert(roffs[t].sub);
655         return((*roffs[t].sub)
656             (r, t, bufp, szp,
657             ln, ppos, pos, offs));
658     }
659
660     /*
661     * Lastly, as we've no scope open, try to look up and execute
662     * the new macro.  If no macro is found, simply return and let
663     * the compilers handle it.
664     */
665
666     if (ROFF_MAX == (t = roff_parse(r, *bufp, &pos)))
667         return(ROFF_CONT);
668
669     assert(roffs[t].proc);
670     return((*roffs[t].proc)
671         (r, t, bufp, szp,
672         ln, ppos, pos, offs));
673 }
674
675 void
676 roff_endparse(struct roff *r)
677 {
678     if (r->last)
679         mandoc_msg(MANDOCERR_SCOPEEXIT, r->parse,
680             r->last->line, r->last->col, NULL);
681
682     if (r->eqn) {
683         mandoc_msg(MANDOCERR_SCOPEEXIT, r->parse,
684             r->eqn->eqn.ln, r->eqn->eqn.pos, NULL);
685         eqn_end(&r->eqn);
686     }
687
688     if (r->tbl) {
689         mandoc_msg(MANDOCERR_SCOPEEXIT, r->parse,
690             r->tbl->line, r->tbl->pos, NULL);
691         tbl_end(&r->tbl);
692     }
693 }
694
695 /*
696 * Parse a roff node's type from the input buffer.  This must be in the
697 * form of ".foo xxx" in the usual way.
698 */
699 static enum roffft
700 roff_parse(struct roff *r, const char *buf, int *pos)
701 {
702     const char *mac;
703     size_t maclen;
704     enum roffft t;
705
706     if ('\0' == buf[*pos] || '"' == buf[*pos] ||
707         '\t' == buf[*pos] || '\'' == buf[*pos])
708         return(ROFF_MAX);
709
710     /*
711     * We stop the macro parse at an escape, tab, space, or nil.
712     * However, '\\' is also a valid macro, so make sure we don't
713     * clobber it by seeing the '\\' as the end of token.
714     */

```

```

715     mac = buf + *pos;
716     maclen = strcspn(mac + 1, "\\t\0") + 1;
717
718     t = (r->current_string = roff_getstrn(r, mac, maclen))
719         ? ROFF_USERDEF : roffhash_find(mac, maclen);
720
721     *pos += (int)maclen;
722
723     while (buf[*pos] && ' ' == buf[*pos])
724         (*pos)++;
725
726     return(t);
727 }
728
729 /* ARGSUSED */
730 static enum rofferr
731 roff_cblock(ROFF_ARGS)
732 {
733     /*
734     * A block-close '..' should only be invoked as a child of an
735     * ignore macro, otherwise raise a warning and just ignore it.
736     */
737
738     if (NULL == r->last) {
739         mandoc_msg(MANDOCERR_NOSCOPE, r->parse, ln, ppos, NULL);
740         return(ROFF_IGN);
741     }
742
743     switch (r->last->tok) {
744     case (ROFF_am):
745         /* FALLTHROUGH */
746     case (ROFF_ami):
747         /* FALLTHROUGH */
748     case (ROFF_aml):
749         /* FALLTHROUGH */
750     case (ROFF_de):
751         /* ROFF_del is remapped to ROFF_de in roff_block(). */
752         /* FALLTHROUGH */
753     case (ROFF_dei):
754         /* FALLTHROUGH */
755     case (ROFF_ig):
756         break;
757     default:
758         mandoc_msg(MANDOCERR_NOSCOPE, r->parse, ln, ppos, NULL);
759         return(ROFF_IGN);
760     }
761
762     if ((*bufp)[pos])
763         mandoc_msg(MANDOCERR_ARGSLIST, r->parse, ln, pos, NULL);
764
765     roffnode_pop(r);
766     roffnode_cleanscope(r);
767     return(ROFF_IGN);
768 }
769
770 static void
771 roffnode_cleanscope(struct roff *r)
772 {
773     while (r->last) {
774         if (--r->last->endspan < 0)
775             break;

```

```

783         roffnode_pop(r);
784     }
785 }

788 /* ARGSUSED */
789 static enum rofferr
790 roff_ccond(ROFF_ARGS)
791 {
793     if (NULL == r->last) {
794         mandoc_msg(MANDOCERR_NOSCOPE, r->parse, ln, ppos, NULL);
795         return(ROFF_IGN);
796     }

798     switch (r->last->tok) {
799     case (ROFF_el):
800         /* FALLTHROUGH */
801     case (ROFF_ie):
802         /* FALLTHROUGH */
803     case (ROFF_if):
804         break;
805     default:
806         mandoc_msg(MANDOCERR_NOSCOPE, r->parse, ln, ppos, NULL);
807         return(ROFF_IGN);
808     }

810     if (r->last->endspan > -1) {
811         mandoc_msg(MANDOCERR_NOSCOPE, r->parse, ln, ppos, NULL);
812         return(ROFF_IGN);
813     }

815     if ((*bufp)[pos])
816         mandoc_msg(MANDOCERR_ARGSLOST, r->parse, ln, pos, NULL);

818     roffnode_pop(r);
819     roffnode_cleanscope(r);
820     return(ROFF_IGN);
821 }

824 /* ARGSUSED */
825 static enum rofferr
826 roff_block(ROFF_ARGS)
827 {
828     int          sv;
829     size_t       sz;
830     char         *name;

832     name = NULL;

834     if (ROFF_ig != tok) {
835         if ('\0' == (*bufp)[pos]) {
836             mandoc_msg(MANDOCERR_NOARGS, r->parse, ln, ppos, NULL);
837             return(ROFF_IGN);
838         }

840         /*
841          * Re-write 'del', since we don't really care about
842          * groff's strange compatibility mode, into 'de'.
843          */

845         if (ROFF_del == tok)
846             tok = ROFF_de;
847         if (ROFF_de == tok)
848             name = *bufp + pos;

```

```

849     else
850         mandoc_msg(MANDOCERR_REQUEST, r->parse, ln, ppos,
851             roffs[tok].name);

853     while ((*bufp)[pos] && ! isspace((unsigned char)(*bufp)[pos]))
854         pos++;

856     while (isspace((unsigned char)(*bufp)[pos]))
857         (*bufp)[pos++] = '\0';
858 }

860     roffnode_push(r, tok, name, ln, ppos);

862     /*
863      * At the beginning of a 'de' macro, clear the existing string
864      * with the same name, if there is one. New content will be
865      * added from roff_block_text() in multiline mode.
866      */

868     if (ROFF_de == tok)
869         roff_setstr(r, name, "", 0);

871     if ('\0' == (*bufp)[pos])
872         return(ROFF_IGN);

874     /* If present, process the custom end-of-line marker. */

876     sv = pos;
877     while ((*bufp)[pos] && ! isspace((unsigned char)(*bufp)[pos]))
878         pos++;

880     /*
881      * Note: groff does NOT like escape characters in the input.
882      * Instead of detecting this, we're just going to let it fly and
883      * to hell with it.
884      */

886     assert(pos > sv);
887     sz = (size_t)(pos - sv);

889     if (1 == sz && '.' == (*bufp)[sv])
890         return(ROFF_IGN);

892     r->last->end = mandoc_malloc(sz + 1);

894     memcpy(r->last->end, *bufp + sv, sz);
895     r->last->end[(int)sz] = '\0';

897     if ((*bufp)[pos])
898         mandoc_msg(MANDOCERR_ARGSLOST, r->parse, ln, pos, NULL);

900     return(ROFF_IGN);
901 }

904 /* ARGSUSED */
905 static enum rofferr
906 roff_block_sub(ROFF_ARGS)
907 {
908     enum roffft   t;
909     int           i, j;

911     /*
912      * First check whether a custom macro exists at this level. If
913      * it does, then check against it. This is some of groff's
914      * stranger behaviours. If we encountered a custom end-scope

```

```

915  * tag and that tag also happens to be a "real" macro, then we
916  * need to try interpreting it again as a real macro.  If it's
917  * not, then return ignore.  Else continue.
918  */
920  if (r->last->end) {
921      for (i = pos, j = 0; r->last->end[j]; j++, i++)
922          if ((*bufp)[i] != r->last->end[j])
923              break;
925      if ('\0' == r->last->end[j] &&
926          ('\0' == (*bufp)[i] ||
927           ' ' == (*bufp)[i] ||
928           '\t' == (*bufp)[i])) {
929          roffnode_pop(r);
930          roffnode_cleanscope(r);
932          while (' ' == (*bufp)[i] || '\t' == (*bufp)[i])
933              i++;
935          pos = i;
936          if (ROFF_MAX != roff_parse(r, *bufp, &pos))
937              return(ROFF_RERUN);
938          return(ROFF_IGN);
939      }
940  }
942  /*
943  * If we have no custom end-query or lookup failed, then try
944  * pulling it out of the hashtable.
945  */
947  t = roff_parse(r, *bufp, &pos);
949  /*
950  * Macros other than block-end are only significant
951  * in 'de' blocks; elsewhere, simply throw them away.
952  */
953  if (ROFF_cblock != t) {
954      if (ROFF_de == tok)
955          roff_setstr(r, r->last->name, *bufp + ppos, 1);
956      return(ROFF_IGN);
957  }
959  assert(roffs[t].proc);
960  return((*roffs[t].proc)(r, t, bufp, szp,
961                        ln, ppos, pos, offs));
962 }
965 /* ARGSUSED */
966 static enum rofferr
967 roff_block_text(ROFF_ARGS)
968 {
970     if (ROFF_de == tok)
971         roff_setstr(r, r->last->name, *bufp + pos, 1);
973     return(ROFF_IGN);
974 }
977 /* ARGSUSED */
978 static enum rofferr
979 roff_cond_sub(ROFF_ARGS)
980 {

```

```

981     enum roffft      t;
982     enum roffrule    rr;
983     char             *ep;
985     rr = r->last->rule;
986     roffnode_cleanscope(r);
988     /*
989     * If the macro is unknown, first check if it contains a closing
990     * delimiter '\}'.  If it does, close out our scope and return
991     * the currently-scoped rule (ignore or continue).  Else, drop
992     * into the currently-scoped rule.
993     */
995     if (ROFF_MAX == (t = roff_parse(r, *bufp, &pos))) {
996         ep = &(*bufp)[pos];
997         for (; NULL != (ep = strchr(ep, '\\')); ep++) {
998             ep++;
999             if ('}' != *ep)
1000                 continue;
1002         /*
1003         * Make the \} go away.
1004         * This is a little haphazard, as it's not quite
1005         * clear how nroff does this.
1006         * If we're at the end of line, then just chop
1007         * off the \} and resize the buffer.
1008         * If we aren't, then conver it to spaces.
1009         */
1011         if ('\0' == *(ep + 1)) {
1012             *--ep = '\0';
1013             *szp -= 2;
1014         } else
1015             *(ep - 1) = *ep = ' ';
1017         roff_ccond(r, ROFF_ccond, bufp, szp,
1018                 ln, pos, pos + 2, offs);
1019         break;
1020     }
1021     return(ROFFRULE_DENY == rr ? ROFF_IGN : ROFF_CONT);
1022 }
1024 /*
1025 * A denied conditional must evaluate its children if and only
1026 * if they're either structurally required (such as loops and
1027 * conditionals) or a closing macro.
1028 */
1030 if (ROFFRULE_DENY == rr)
1031     if (! (ROFFMAC_STRUCT & roffs[t].flags))
1032         if (ROFF_ccond != t)
1033             return(ROFF_IGN);
1035 assert(roffs[t].proc);
1036 return((*roffs[t].proc)(r, t, bufp, szp,
1037                        ln, ppos, pos, offs));
1038 }
1040 /* ARGSUSED */
1041 static enum rofferr
1042 roff_cond_text(ROFF_ARGS)
1043 {
1044     char             *ep;
1045     enum roffrule    rr;

```

```

1047     rr = r->last->rule;
1048     roffnode_cleanscope(r);

1050     ep = &(*bufp)[pos];
1051     for ( ; NULL != (ep = strchr(ep, '\\')); ep++) {
1052         ep++;
1053         if ('}' != *ep)
1054             continue;
1055         *ep = '&';
1056         roff_ccond(r, ROFF_ccond, bufp, szp,
1057                 ln, pos, pos + 2, ofs);
1058     }
1059     return(ROFFRULE_DENY == rr ? ROFF_IGN : ROFF_CONT);
1060 }

1062 static enum roffrule
1063 roff_evalcond(const char *v, int *pos)
1064 {

1066     switch (v[*pos]) {
1067     case ('n'):
1068         (*pos)++;
1069         return(ROFFRULE_ALLOW);
1070     case ('e'):
1071         /* FALLTHROUGH */
1072     case ('o'):
1073         /* FALLTHROUGH */
1074     case ('t'):
1075         (*pos)++;
1076         return(ROFFRULE_DENY);
1077     default:
1078         break;
1079     }

1081     while (v[*pos] && ' ' != v[*pos])
1082         (*pos)++;
1083     return(ROFFRULE_DENY);
1084 }

1086 /* ARGSUSED */
1087 static enum rofferr
1088 roff_line_ignore(ROFF_ARGS)
1089 {

1091     if (ROFF_it == tok)
1092         mandoc_msg(MANDOCERR_REQUEST, r->parse, ln, ppos, "it");

1094     return(ROFF_IGN);
1095 }

1097 /* ARGSUSED */
1098 static enum rofferr
1099 roff_cond(ROFF_ARGS)
1100 {
1101     int             sv;
1102     enum roffrule  rule;

1104     /*
1105      * An `el' has no conditional body: it will consume the value
1106      * of the current rstack entry set in prior `ie' calls or
1107      * defaults to DENY.
1108      *
1109      * If we're not an `el', however, then evaluate the conditional.
1110      */

1112     rule = ROFF_el == tok ?

```

```

1113         (r->rstackpos < 0 ?
1114          ROFFRULE_DENY : r->rstack[r->rstackpos--]) :
1115         roff_evalcond(*bufp, &pos);

1117     sv = pos;
1118     while (' ' == (*bufp)[pos])
1119         pos++;

1121     /*
1122      * Roff is weird. If we have just white-space after the
1123      * conditional, it's considered the BODY and we exit without
1124      * really doing anything. Warn about this. It's probably
1125      * wrong.
1126      */

1128     if ('\0' == (*bufp)[pos] && sv != pos) {
1129         mandoc_msg(MANDOCERR_NOARGS, r->parse, ln, ppos, NULL);
1130         return(ROFF_IGN);
1131     }

1133     roffnode_push(r, tok, NULL, ln, ppos);

1135     r->last->rule = rule;

1137     /*
1138      * An if-else will put the NEGATION of the current evaluated
1139      * conditional into the stack of rules.
1140      */

1142     if (ROFF_ie == tok) {
1143         if (r->rstackpos == RSTACK_MAX - 1) {
1144             mandoc_msg(MANDOCERR_MEM,
1145                 r->parse, ln, ppos, NULL);
1146             return(ROFF_ERR);
1147         }
1148         r->rstack[++r->rstackpos] =
1149             ROFFRULE_DENY == r->last->rule ?
1150             ROFFRULE_ALLOW : ROFFRULE_DENY;
1151     }

1153     /* If the parent has false as its rule, then so do we. */

1155     if (r->last->parent && ROFFRULE_DENY == r->last->parent->rule)
1156         r->last->rule = ROFFRULE_DENY;

1158     /*
1159      * Determine scope. If we're invoked with "{ " trailing the
1160      * conditional, then we're in a multiline scope. Else our scope
1161      * expires on the next line.
1162      */

1164     r->last->endspan = 1;

1166     if ('\\' == (*bufp)[pos] && '{' == (*bufp)[pos + 1]) {
1167         r->last->endspan = -1;
1168         pos += 2;
1169     }

1171     /*
1172      * If there are no arguments on the line, the next-line scope is
1173      * assumed.
1174      */

1176     if ('\0' == (*bufp)[pos])
1177         return(ROFF_IGN);

```

```

1179      /* Otherwise re-run the roff parser after recalculating. */
1181      *offs = pos;
1182      return(ROFF_RERUN);
1183 }

1186 /* ARGSUSED */
1187 static enum rofferr
1188 roff_ds(ROFF_ARGS)
1189 {
1190     char          *name, *string;

1192     /*
1193      * A symbol is named by the first word following the macro
1194      * invocation up to a space. Its value is anything after the
1195      * name's trailing whitespace and optional double-quote. Thus,
1196      *
1197      * [.ds foo "bar " ]
1198      *
1199      * will have 'bar " ' as its value.
1200      */

1202     string = *bufp + pos;
1203     name = roff_getname(r, &string, ln, pos);
1204     if ('\0' == *name)
1205         return(ROFF_IGN);

1207     /* Read past initial double-quote. */
1208     if ("'" == *string)
1209         string++;

1211     /* The rest is the value. */
1212     roff_setstr(r, name, string, 0);
1213     return(ROFF_IGN);
1214 }

1216 int
1217 roff_regisset(const struct roff *r, enum regs reg)
1218 {
1220     return(r->regs[(int)reg].set);
1221 }

1223 unsigned int
1224 roff_regget(const struct roff *r, enum regs reg)
1225 {
1227     return(r->regs[(int)reg].u);
1228 }

1230 void
1231 roff_regunset(struct roff *r, enum regs reg)
1232 {
1234     r->regs[(int)reg].set = 0;
1235 }

1237 /* ARGSUSED */
1238 static enum rofferr
1239 roff_nr(ROFF_ARGS)
1240 {
1241     const char    *key;
1242     char          *val;
1243     int           iv;

```

```

1245     val = *bufp + pos;
1246     key = roff_getname(r, &val, ln, pos);

1248     if (0 == strcmp(key, "nS")) {
1249         r->regs[(int)REG_NS].set = 1;
1250         if ((iv = mandoc_strntoi(val, strlen(val), 10)) >= 0)
1251             r->regs[(int)REG_NS].u = (unsigned)iv;
1252         else
1253             r->regs[(int)REG_NS].u = 0u;
1254     }

1256     return(ROFF_IGN);
1257 }

1259 /* ARGSUSED */
1260 static enum rofferr
1261 roff_rm(ROFF_ARGS)
1262 {
1263     const char    *name;
1264     char          *cp;

1266     cp = *bufp + pos;
1267     while ('\0' != *cp) {
1268         name = roff_getname(r, &cp, ln, (int)(cp - *bufp));
1269         if ('\0' != *name)
1270             roff_setstr(r, name, NULL, 0);
1271     }
1272     return(ROFF_IGN);
1273 }

1275 /* ARGSUSED */
1276 static enum rofferr
1277 roff_TE(ROFF_ARGS)
1278 {
1280     if (NULL == r->tbl)
1281         mandoc_msg(MANDOCERR_NOSCOPE, r->parse, ln, ppos, NULL);
1282     else
1283         tbl_end(&r->tbl);

1285     return(ROFF_IGN);
1286 }

1288 /* ARGSUSED */
1289 static enum rofferr
1290 roff_T_(ROFF_ARGS)
1291 {
1293     if (NULL == r->tbl)
1294         mandoc_msg(MANDOCERR_NOSCOPE, r->parse, ln, ppos, NULL);
1295     else
1296         tbl_restart(ppos, ln, r->tbl);

1298     return(ROFF_IGN);
1299 }

1301 #if 0
1302 static int
1303 roff_closeeqn(struct roff *r)
1304 {
1306     return(r->eqn && ROFF_EQN == eqn_end(&r->eqn) ? 1 : 0);
1307 }
1308 #endif

1310 static void

```

```

1311 roff_openeqn(struct roff *r, const char *name, int line,
1312             int offs, const char *buf)
1313 {
1314     struct eqn_node *e;
1315     int             poff;

1317     assert(NULL == r->eqn);
1318     e = eqn_alloc(name, offs, line, r->parse);

1320     if (r->last_eqn)
1321         r->last_eqn->next = e;
1322     else
1323         r->first_eqn = r->last_eqn = e;

1325     r->eqn = r->last_eqn = e;

1327     if (buf) {
1328         poff = 0;
1329         eqn_read(&r->eqn, line, buf, offs, &poff);
1330     }
1331 }

1333 /* ARGSUSED */
1334 static enum rofferr
1335 roff_EQ(ROFF_ARGS)
1336 {
1338     roff_openeqn(r, *bufp + pos, ln, ppos, NULL);
1339     return(ROFF_IGN);
1340 }

1342 /* ARGSUSED */
1343 static enum rofferr
1344 roff_EN(ROFF_ARGS)
1345 {
1347     mandoc_msg(MANDOCERR_NOSCOPE, r->parse, ln, ppos, NULL);
1348     return(ROFF_IGN);
1349 }

1351 /* ARGSUSED */
1352 static enum rofferr
1353 roff_TS(ROFF_ARGS)
1354 {
1355     struct tbl_node *t;

1357     if (r->tbl) {
1358         mandoc_msg(MANDOCERR_SCOPEBROKEN, r->parse, ln, ppos, NULL);
1359         tbl_end(&r->tbl);
1360     }

1362     t = tbl_alloc(ppos, ln, r->parse);

1364     if (r->last_tbl)
1365         r->last_tbl->next = t;
1366     else
1367         r->first_tbl = r->last_tbl = t;

1369     r->tbl = r->last_tbl = t;
1370     return(ROFF_IGN);
1371 }

1373 /* ARGSUSED */
1374 static enum rofferr
1375 roff_tr(ROFF_ARGS)
1376 {

```

```

1377     const char      *p, *first, *second;
1378     size_t          fsz, ssz;
1379     enum mandoc_esc esc;

1381     p = *bufp + pos;

1383     if ('\0' == *p) {
1384         mandoc_msg(MANDOCERR_ARGCOUNT, r->parse, ln, ppos, NULL);
1385         return(ROFF_IGN);
1386     }

1388     while ('\0' != *p) {
1389         fsz = ssz = 1;

1391         first = p++;
1392         if ('\\" == *first) {
1393             esc = mandoc_escape(&p, NULL, NULL);
1394             if (ESCAPE_ERROR == esc) {
1395                 mandoc_msg
1396                     (MANDOCERR_BADESCAPE, r->parse,
1397                      ln, (int)(p - *bufp), NULL);
1398                 return(ROFF_IGN);
1399             }
1400             fsz = (size_t)(p - first);
1401         }

1403         second = p++;
1404         if ('\\" == *second) {
1405             esc = mandoc_escape(&p, NULL, NULL);
1406             if (ESCAPE_ERROR == esc) {
1407                 mandoc_msg
1408                     (MANDOCERR_BADESCAPE, r->parse,
1409                      ln, (int)(p - *bufp), NULL);
1410                 return(ROFF_IGN);
1411             }
1412             ssz = (size_t)(p - second);
1413         } else if ('\0' == *second) {
1414             mandoc_msg(MANDOCERR_ARGCOUNT, r->parse,
1415                        ln, (int)(p - *bufp), NULL);
1416             second = " ";
1417             p--;
1418         }

1420         if (fsz > 1) {
1421             roff_setstrn(&r->xmbtab, first,
1422                          fsz, second, ssz, 0);
1423             continue;
1424         }

1426         if (NULL == r->xtab)
1427             r->xtab = mandoc_calloc
1428                 (128, sizeof(struct roffstr));

1430         free(r->xtab[(int)*first].p);
1431         r->xtab[(int)*first].p = mandoc_strdup(second, ssz);
1432         r->xtab[(int)*first].sz = ssz;
1433     }

1435     return(ROFF_IGN);
1436 }

1438 /* ARGSUSED */
1439 static enum rofferr
1440 roff_so(ROFF_ARGS)
1441 {
1442     char *name;

```



```

1444     mandoc_msg(MANDOCERR_SO, r->parse, ln, ppos, NULL);
1446     /*
1447     * Handle 'so'. Be EXTREMELY careful, as we shouldn't be
1448     * opening anything that's not in our cwd or anything beneath
1449     * it. Thus, explicitly disallow traversing up the file-system
1450     * or using absolute paths.
1451     */
1453     name = *bufp + pos;
1454     if ('/' == *name || strstr(name, "../") || strstr(name, "/..")) {
1455         mandoc_msg(MANDOCERR_SOPATH, r->parse, ln, pos, NULL);
1456         return(ROFF_ERR);
1457     }
1459     *offs = pos;
1460     return(ROFF_SO);
1461 }
1463 /* ARGSUSED */
1464 static enum rofferr
1465 roff_userdef(ROFF_ARGS)
1466 {
1467     const char      *arg[9];
1468     char            *cp, *nl, *n2;
1469     int              i;
1471     /*
1472     * Collect pointers to macro argument strings
1473     * and null-terminate them.
1474     */
1475     cp = *bufp + pos;
1476     for (i = 0; i < 9; i++)
1477         arg[i] = '\0' == *cp ? "" :
1478             mandoc_getarg(r->parse, &cp, ln, &pos);
1480     /*
1481     * Expand macro arguments.
1482     */
1483     *szp = 0;
1484     nl = cp = mandoc_strdup(r->current_string);
1485     while (NULL != (cp = strstr(cp, "\\$"))) {
1486         i = cp[2] - '1';
1487         if (0 > i || 8 < i) {
1488             /* Not an argument invocation. */
1489             cp += 2;
1490             continue;
1491         }
1493         *szp = strlen(nl) - 3 + strlen(arg[i]) + 1;
1494         n2 = mandoc_malloc(*szp);
1496         strncpy(n2, nl, (size_t)(cp - nl + 1));
1497         strlcat(n2, arg[i], *szp);
1498         strlcat(n2, cp + 3, *szp);
1500         cp = n2 + (cp - nl);
1501         free(nl);
1502         nl = n2;
1503     }
1505     /*
1506     * Replace the macro invocation
1507     * by the expanded macro.
1508     */

```

```

1509     free(*bufp);
1510     *bufp = nl;
1511     if (0 == *szp)
1512         *szp = strlen(*bufp) + 1;
1514     return(*szp > 1 && '\n' == (*bufp)[(int)*szp - 2] ?
1515         ROFF_REPARSE : ROFF_APPEND);
1516 }
1518 static char *
1519 roff_getname(struct roff *r, char **cpp, int ln, int pos)
1520 {
1521     char      *name, *cp;
1523     name = *cpp;
1524     if ('\0' == *name)
1525         return(name);
1527     /* Read until end of name. */
1528     for (cp = name; '\0' != *cp && ' ' != *cp; cp++) {
1529         if ('\ ' != *cp)
1530             continue;
1531         cp++;
1532         if ('\ ' == *cp)
1533             continue;
1534         mandoc_msg(MANDOCERR_NAMESC, r->parse, ln, pos, NULL);
1535         *cp = '\0';
1536         name = cp;
1537     }
1539     /* Nil-terminate name. */
1540     if ('\0' != *cp)
1541         *(cp++) = '\0';
1543     /* Read past spaces. */
1544     while (' ' == *cp)
1545         cp++;
1547     *cpp = cp;
1548     return(name);
1549 }
1551 /*
1552 * Store *string into the user-defined string called *name.
1553 * In multiline mode, append to an existing entry and append '\n';
1554 * else replace the existing entry, if there is one.
1555 * To clear an existing entry, call with (*r, *name, NULL, 0).
1556 */
1557 static void
1558 roff_setstr(struct roff *r, const char *name, const char *string,
1559             int multiline)
1560 {
1562     roff_setstrn(&r->strtab, name, strlen(name), string,
1563                 string ? strlen(string) : 0, multiline);
1564 }
1566 static void
1567 roff_setstrn(struct roffkv **r, const char *name, size_t namesz,
1568              const char *string, size_t stringsz, int multiline)
1569 {
1570     struct roffkv *n;
1571     char           *c;
1572     int            i;
1573     size_t         oldch, newch;

```

```

1575 /* Search for an existing string with the same name. */
1576 n = *r;

1578 while (n && strcmp(name, n->key.p))
1579     n = n->next;

1581 if (NULL == n) {
1582     /* Create a new string table entry. */
1583     n = mandoc_malloc(sizeof(struct roffkv));
1584     n->key.p = mandoc_strdup(name, namesz);
1585     n->key.sz = namesz;
1586     n->val.p = NULL;
1587     n->val.sz = 0;
1588     n->next = *r;
1589     *r = n;
1590 } else if (0 == multiline) {
1591     /* In multiline mode, append; else replace. */
1592     free(n->val.p);
1593     n->val.p = NULL;
1594     n->val.sz = 0;
1595 }

1597 if (NULL == string)
1598     return;

1600 /*
1601  * One additional byte for the '\n' in multiline mode,
1602  * and one for the terminating '\0'.
1603  */
1604 newch = stringsz + (multiline ? 2u : 1u);

1606 if (NULL == n->val.p) {
1607     n->val.p = mandoc_malloc(newch);
1608     *n->val.p = '\0';
1609     oldch = 0;
1610 } else {
1611     oldch = n->val.sz;
1612     n->val.p = mandoc_realloc(n->val.p, oldch + newch);
1613 }

1615 /* Skip existing content in the destination buffer. */
1616 c = n->val.p + (int)oldch;

1618 /* Append new content to the destination buffer. */
1619 i = 0;
1620 while (i < (int)stringsz) {
1621     /*
1622      * Rudimentary roff copy mode:
1623      * Handle escaped backslashes.
1624      */
1625     if ('\\" == string[i] && '\\\" == string[i + 1])
1626         i++;
1627     *c++ = string[i++];
1628 }

1630 /* Append terminating bytes. */
1631 if (multiline)
1632     *c++ = '\n';

1634 *c = '\0';
1635 n->val.sz = (int)(c - n->val.p);
1636 }

1638 static const char *
1639 roff_getstrn(const struct roff *r, const char *name, size_t len)
1640 {

```

```

1641     const struct roffkv *n;

1643     for (n = r->strtab; n; n = n->next)
1644         if (0 == strncmp(name, n->key.p, len) &&
1645             '\0' == n->key.p[(int)len])
1646             return(n->val.p);

1648     return(NULL);
1649 }

1651 static void
1652 roff_freestr(struct roffkv *r)
1653 {
1654     struct roffkv *n, *nn;

1656     for (n = r; n; n = nn) {
1657         free(n->key.p);
1658         free(n->val.p);
1659         nn = n->next;
1660         free(n);
1661     }
1662 }

1664 const struct tbl_span *
1665 roff_span(const struct roff *r)
1666 {
1667     return(r->tbl ? tbl_span(r->tbl) : NULL);
1668 }

1671 const struct eqn *
1672 roff_eqn(const struct roff *r)
1673 {
1674     return(r->last_eqn ? &r->last_eqn->eqn : NULL);
1675 }

1678 /*
1679  * Duplicate an input string, making the appropriate character
1680  * conversions (as stipulated by 'tr') along the way.
1681  * Returns a heap-allocated string with all the replacements made.
1682  */
1683 char *
1684 roff_strdup(const struct roff *r, const char *p)
1685 {
1686     const struct roffkv *cp;
1687     char *res;
1688     const char *pp;
1689     size_t sz, sz2;
1690     enum mandoc_esc esc;

1692     if (NULL == r->xmbtab && NULL == r->xtab)
1693         return(mandoc_strdup(p));
1694     else if ('\0' == *p)
1695         return(mandoc_strdup(""));

1697     /*
1698      * Step through each character looking for term matches
1699      * (remember that a 'tr' can be invoked with an escape, which is
1700      * a glyph but the escape is multi-character).
1701      * We only do this if the character hash has been initialised
1702      * and the string is >0 length.
1703      */

1705     res = NULL;
1706     sz2 = 0;

```

```

1708     while ('\0' != *p) {
1709         if ('\\" != *p && r->xtab && r->xtab[(int)*p].p) {
1710             sz = r->xtab[(int)*p].sz;
1711             res = mandoc_realloc(res, ssz + sz + 1);
1712             memcpy(res + ssz, r->xtab[(int)*p].p, sz);
1713             ssz += sz;
1714             p++;
1715             continue;
1716         } else if ('\\" != *p) {
1717             res = mandoc_realloc(res, ssz + 2);
1718             res[ssz++] = *p++;
1719             continue;
1720         }
1721
1722         /* Search for term matches. */
1723         for (cp = r->xmbtab; cp; cp = cp->next)
1724             if (0 == strcmp(p, cp->key.p, cp->key.sz))
1725                 break;
1726
1727         if (NULL != cp) {
1728             /*
1729              * A match has been found.
1730              * Append the match to the array and move
1731              * forward by its keysize.
1732              */
1733             res = mandoc_realloc
1734                 (res, ssz + cp->val.sz + 1);
1735             memcpy(res + ssz, cp->val.p, cp->val.sz);
1736             ssz += cp->val.sz;
1737             p += (int)cp->key.sz;
1738             continue;
1739         }
1740
1741         /*
1742          * Handle escapes carefully: we need to copy
1743          * over just the escape itself, or else we might
1744          * do replacements within the escape itself.
1745          * Make sure to pass along the bogus string.
1746          */
1747         pp = p++;
1748         esc = mandoc_escape(&p, NULL, NULL);
1749         if (ESCAPE_ERROR == esc) {
1750             sz = strlen(pp);
1751             res = mandoc_realloc(res, ssz + sz + 1);
1752             memcpy(res + ssz, pp, sz);
1753             break;
1754         }
1755         /*
1756          * We bail out on bad escapes.
1757          * No need to warn: we already did so when
1758          * roff_res() was called.
1759          */
1760         sz = (int)(p - pp);
1761         res = mandoc_realloc(res, ssz + sz + 1);
1762         memcpy(res + ssz, pp, sz);
1763         ssz += sz;
1764     }
1765
1766     res[(int)ssz] = '\0';
1767     return(res);
1768 }

```

new/usr/src/cmd/mandoc/st.c

1

1152 Sat Jul 19 19:24:32 2014

new/usr/src/cmd/mandoc/st.c

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /*      $Id: st.c,v 1.9 2011/03/22 14:33:05 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <stdlib.h>
22 #include <string.h>
23 #include <time.h>
24
25 #include "mdoc.h"
26 #include "mandoc.h"
27 #include "libmdoc.h"
28
29 #define LINE(x, y) \
30     if (0 == strcmp(p, x)) return(y);
31
32 const char *
33 mdoc_a2st(const char *p)
34 {
35
36 #include "st.in"
37
38     return(NULL);
39 }
```

new/usr/src/cmd/mandoc/st.in

1

4542 Sat Jul 19 19:24:32 2014

new/usr/src/cmd/mandoc/st.in

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /* $Id: st.in,v 1.19 2012/02/26 21:47:09 schwarze Exp $ */
2 /*
3  * Copyright (c) 2009, 2010 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17
18 /*
19 * This file defines the .St macro arguments.  If you add a new
20 * standard, make sure that the left-and side corresponds to the .St
21 * argument (like .St -p1003.1) and the right-hand side corresponds to
22 * the formatted output string.
23 *
24 * Be sure to escape strings.
25 * The non-breaking blanks prevent ending an output line right before
26 * a number.  Groff prevent line breaks at the same places.
27 *
28 * REMEMBER TO ADD NEW STANDARDS TO MDOC.7!
29 */
30
31 LINE("-p1003.1-88", "IEEE Std 1003.1-1988 (\\(lqPOSIX.1\\(rq)")
32 LINE("-p1003.1-90", "IEEE Std 1003.1-1990 (\\(lqPOSIX.1\\(rq)")
33 LINE("-p1003.1-96", "ISO/IEC 9945-1:1996 (\\(lqPOSIX.1\\(rq)")
34 LINE("-p1003.1-2001", "IEEE Std 1003.1-2001 (\\(lqPOSIX.1\\(rq)")
35 LINE("-p1003.1-2004", "IEEE Std 1003.1-2004 (\\(lqPOSIX.1\\(rq)")
36 LINE("-p1003.1-2008", "IEEE Std 1003.1-2008 (\\(lqPOSIX.1\\(rq)")
37 LINE("-p1003.1", "IEEE Std 1003.1 (\\(lqPOSIX.1\\(rq)")
38 LINE("-p1003.1b", "IEEE Std 1003.1b (\\(lqPOSIX.1\\(rq)")
39 LINE("-p1003.1b-93", "IEEE Std 1003.1b-1993 (\\(lqPOSIX.1\\(rq)")
40 LINE("-p1003.1c-95", "IEEE Std 1003.1c-1995 (\\(lqPOSIX.1\\(rq)")
41 LINE("-p1003.1g-2000", "IEEE Std 1003.1g-2000 (\\(lqPOSIX.1\\(rq)")
42 LINE("-p1003.1i-95", "IEEE Std 1003.1i-1995 (\\(lqPOSIX.1\\(rq)")
43 LINE("-p1003.2-92", "IEEE Std 1003.2-1992 (\\(lqPOSIX.2\\(rq)")
44 LINE("-p1003.2a-92", "IEEE Std 1003.2a-1992 (\\(lqPOSIX.2\\(rq)")
45 LINE("-p1387.2-95", "IEEE Std 1387.2-1995 (\\(lqPOSIX.7.2\\(rq)")
46 LINE("-p1003.2", "IEEE Std 1003.2 (\\(lqPOSIX.2\\(rq)")
47 LINE("-p1387.2", "IEEE Std 1387.2 (\\(lqPOSIX.7.2\\(rq)")
48 LINE("-isoC", "ISO/IEC 9899:1990 (\\(lqISO\\(rq)")
49 LINE("-isoC-90", "ISO/IEC 9899:1990 (\\(lqISO\\(rq)")
50 LINE("-isoC-amd1", "ISO/IEC 9899/AMD1:1995 (\\(lqISO\\(rq)", Amendment 1\\(r
51 LINE("-isoC-tcor1", "ISO/IEC 9899/TCOR1:1994 (\\(lqISO\\(rq)", Technical Corr
52 LINE("-isoC-tcor2", "ISO/IEC 9899/TCOR2:1995 (\\(lqISO\\(rq)", Technical Corr
53 LINE("-isoC-99", "ISO/IEC 9899:1999 (\\(lqISO\\(rq)")
54 LINE("-isoC-2011", "ISO/IEC 9899:2011 (\\(lqISO\\(rq)")
55 LINE("-iso9945-1-90", "ISO/IEC 9945-1:1990 (\\(lqPOSIX.1\\(rq)")
56 LINE("-iso9945-1-96", "ISO/IEC 9945-1:1996 (\\(lqPOSIX.1\\(rq)"/>
```

new/usr/src/cmd/mandoc/st.in

2

```
57 LINE("-iso9945-2-93", "ISO/IEC 9945-2:1993 (\\(lqPOSIX.2\\(rq)")
58 LINE("-ansiC", "ANSI X3.159-1989 (\\(lqANSI\\(rq)")
59 LINE("-ansiC-89", "ANSI X3.159-1989 (\\(lqANSI\\(rq)")
60 LINE("-ansiC-99", "ANSI/ISO/IEC 9899-1999 (\\(lqANSI\\(rq)")
61 LINE("-ieee754", "IEEE Std 754-1985")
62 LINE("-iso8802-3", "ISO 8802-3: 1989")
63 LINE("-iso8601", "ISO 8601")
64 LINE("-ieee1275-94", "IEEE Std 1275-1994 (\\(lqOpen Firmware\\(rq)")
65 LINE("-xpg3", "X/Open Portability Guide Issue\\(rq)", 3 (\\(lqXPG3\\(rq)")
66 LINE("-xpg4", "X/Open Portability Guide Issue\\(rq)", 4 (\\(lqXPG4\\(rq)")
67 LINE("-xpg4.2", "X/Open Portability Guide Issue\\(rq)", 4, Version\\(rq)", 2 (\\(lqX
68 LINE("-xpg4.3", "X/Open Portability Guide Issue\\(rq)", 4, Version\\(rq)", 3 (\\(lqX
69 LINE("-xbd5", "X/Open Base Definitions Issue\\(rq)", 5 (\\(lqXBD5\\(rq)")
70 LINE("-xcu5", "X/Open Commands and Utilities Issue\\(rq)", 5 (\\(lqXCU5\\(rq
71 LINE("-xsh5", "X/Open System Interfaces and Headers Issue\\(rq)", 5 (\\(lqXS
72 LINE("-xns5", "X/Open Networking Services Issue\\(rq)", 5 (\\(lqXNS5\\(rq)")
73 LINE("-xns5.2", "X/Open Networking Services Issue\\(rq)", 5.2 (\\(lqXNS5.2\\(r
74 LINE("-xns5.2d2.0", "X/Open Networking Services Issue\\(rq)", 5.2 Draft\\(rq)", 2.0 (\\(
75 LINE("-xcurses4.2", "X/Open Curses Issue\\(rq)", 4, Version\\(rq)", 2 (\\(lqXCURSES4.2\\(
76 LINE("-susv2", "Version\\(rq)", 2 of the Single UNIX Specification")
77 LINE("-susv3", "Version\\(rq)", 3 of the Single UNIX Specification")
78 LINE("-svid4", "System\\(rq)", V Interface Definition, Fourth Edition (\\(lqS"/>
```

```

*****
4049 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/tbl.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: tbl.c,v 1.26 2011/07/25 15:37:00 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <assert.h>
23 #include <stdio.h>
24 #include <stdlib.h>
25 #include <string.h>
26 #include <time.h>
27
28 #include "mandoc.h"
29 #include "libmandoc.h"
30 #include "libroff.h"
31
32 enum rofferr
33 tbl_read(struct tbl_node *tbl, int ln, const char *p, int offs)
34 {
35     int          len;
36     const char  *cp;
37
38     cp = &p[offs];
39     len = (int)strlen(cp);
40
41     /*
42      * If we're in the options section and we don't have a
43      * terminating semicolon, assume we've moved directly into the
44      * layout section. No need to report a warning: this is,
45      * apparently, standard behaviour.
46      */
47
48     if (TBL_PART_OPTS == tbl->part && len)
49         if (';' != cp[len - 1])
50             tbl->part = TBL_PART_LAYOUT;
51
52     /* Now process each logical section of the table.  */
53
54     switch (tbl->part) {
55     case (TBL_PART_OPTS):
56         return(tbl_option(tbl, ln, p) ? ROFF_IGN : ROFF_ERR);

```

```

57     case (TBL_PART_LAYOUT):
58         return(tbl_layout(tbl, ln, p) ? ROFF_IGN : ROFF_ERR);
59     case (TBL_PART_CDATA):
60         return(tbl_cdata(tbl, ln, p) ? ROFF_TBL : ROFF_IGN);
61     default:
62         break;
63     }
64
65     /*
66      * This only returns zero if the line is empty, so we ignore it
67      * and continue on.
68      */
69     return(tbl_data(tbl, ln, p) ? ROFF_TBL : ROFF_IGN);
70 }
71
72 struct tbl_node *
73 tbl_alloc(int pos, int line, struct mparse *parse)
74 {
75     struct tbl_node *p;
76
77     p = mandoc_calloc(1, sizeof(struct tbl_node));
78     p->line = line;
79     p->pos = pos;
80     p->parse = parse;
81     p->part = TBL_PART_OPTS;
82     p->opts.tab = '\t';
83     p->opts.linesize = 12;
84     p->opts.decimal = '.';
85     return(p);
86 }
87
88 void
89 tbl_free(struct tbl_node *p)
90 {
91     struct tbl_row *rp;
92     struct tbl_cell *cp;
93     struct tbl_span *sp;
94     struct tbl_dat *dp;
95     struct tbl_head *hp;
96
97     while (NULL != (rp = p->first_row)) {
98         p->first_row = rp->next;
99         while (rp->first) {
100             cp = rp->first;
101             rp->first = cp->next;
102             free(cp);
103         }
104         free(rp);
105     }
106
107     while (NULL != (sp = p->first_span)) {
108         p->first_span = sp->next;
109         while (sp->first) {
110             dp = sp->first;
111             sp->first = dp->next;
112             if (dp->string)
113                 free(dp->string);
114             free(dp);
115         }
116         free(sp);
117     }
118
119     while (NULL != (hp = p->first_head)) {
120         p->first_head = hp->next;
121         free(hp);
122     }

```

```
124     free(p);
125 }

127 void
128 tbl_restart(int line, int pos, struct tbl_node *tbl)
129 {
130     if (TBL_PART_CDATA == tbl->part)
131         mandoc_msg(MANDOCERR_TBLBLOCK, tbl->parse,
132                 tbl->line, tbl->pos, NULL);

134     tbl->part = TBL_PART_LAYOUT;
135     tbl->line = line;
136     tbl->pos = pos;

138     if (NULL == tbl->first_span || NULL == tbl->first_span->first)
139         mandoc_msg(MANDOCERR_TBLNODATA, tbl->parse,
140                 tbl->line, tbl->pos, NULL);
141 }

143 const struct tbl_span *
144 tbl_span(struct tbl_node *tbl)
145 {
146     struct tbl_span *span;

148     assert(tbl);
149     span = tbl->current_span ? tbl->current_span->next
150         : tbl->first_span;
151     if (span)
152         tbl->current_span = span;
153     return(span);
154 }

156 void
157 tbl_end(struct tbl_node **tblp)
158 {
159     struct tbl_node *tbl;

161     tbl = *tblp;
162     *tblp = NULL;

164     if (NULL == tbl->first_span || NULL == tbl->first_span->first)
165         mandoc_msg(MANDOCERR_TBLNODATA, tbl->parse,
166                 tbl->line, tbl->pos, NULL);

168     if (tbl->last_span)
169         tbl->last_span->flags |= TBL_SPAN_LAST;

171     if (TBL_PART_CDATA == tbl->part)
172         mandoc_msg(MANDOCERR_TBLBLOCK, tbl->parse,
173                 tbl->line, tbl->pos, NULL);
174 }
```

```

*****
6446 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/tbl_data.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: tbl_data.c,v 1.24 2011/03/20 16:02:05 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <assert.h>
23 #include <ctype.h>
24 #include <stdlib.h>
25 #include <string.h>
26 #include <time.h>
27
28 #include "mandoc.h"
29 #include "libmandoc.h"
30 #include "libroff.h"
31
32 static int
33 data(struct tbl_node *, struct tbl_span *,
34       int, const char *, int *);
35
36 static struct tbl_span *
37 newspan(struct tbl_node *, int,
38         struct tbl_row *);
39
40 static int
41 data(struct tbl_node *tbl, struct tbl_span *dp,
42       int ln, const char *p, int *pos)
43 {
44     struct tbl_dat *dat;
45     struct tbl_cell *cp;
46     int sv, spans;
47
48     cp = NULL;
49     if (dp->last && dp->last->layout)
50         cp = dp->last->layout->next;
51     else if (NULL == dp->last)
52         cp = dp->layout->first;
53
54     /*
55      * Skip over spanners and vertical lines to data formats, since
56      * we want to match data with data layout cells in the header.
57      */
58     while (cp && (TBL_CELL_VERT == cp->pos ||

```

```

57         TBL_CELL_DVERT == cp->pos ||
58         TBL_CELL_SPAN == cp->pos))
59         cp = cp->next;
60
61     /*
62      * Stop processing when we reach the end of the available layout
63      * cells. This means that we have extra input.
64      */
65
66     if (NULL == cp) {
67         mandoc_msg(MANDOCERR_TBLEXTRADAT,
68                  tbl->parse, ln, *pos, NULL);
69         /* Skip to the end... */
70         while (p[*pos])
71             (*pos)++;
72         return(1);
73     }
74
75     dat = mandoc_malloc(1, sizeof(struct tbl_dat));
76     dat->layout = cp;
77     dat->pos = TBL_DATA_NONE;
78
79     assert(TBL_CELL_SPAN != cp->pos);
80
81     for (spans = 0, cp = cp->next; cp; cp = cp->next)
82         if (TBL_CELL_SPAN == cp->pos)
83             spans++;
84         else
85             break;
86
87     dat->spans = spans;
88
89     if (dp->last) {
90         dp->last->next = dat;
91         dp->last = dat;
92     } else
93         dp->last = dp->first = dat;
94
95     sv = *pos;
96     while (p[*pos] && p[*pos] != tbl->opts.tab)
97         (*pos)++;
98
99     /*
100      * Check for a continued-data scope opening. This consists of a
101      * trailing 'T{' at the end of the line. Subsequent lines,
102      * until a standalone 'T}', are included in our cell.
103      */
104
105     if (*pos - sv == 2 && 'T' == p[sv] && '{' == p[sv + 1]) {
106         tbl->part = TBL_PART_CDATA;
107         return(0);
108     }
109
110     assert(*pos - sv >= 0);
111
112     dat->string = mandoc_malloc((size_t)(*pos - sv + 1));
113     memcpy(dat->string, &p[sv], (size_t)(*pos - sv));
114     dat->string[*pos - sv] = '\0';
115
116     if (p[*pos])
117         (*pos)++;
118
119     if (! strcmp(dat->string, "_"))
120         dat->pos = TBL_DATA_HORIZ;
121     else if (! strcmp(dat->string, "="))
122         dat->pos = TBL_DATA_DHORIZ;

```



```

123     else if ( ! strcmp(dat->string, "\\_")
124               dat->pos = TBL_DATA_NHORIZ;
125     else if ( ! strcmp(dat->string, "\\="))
126               dat->pos = TBL_DATA_NDHORIZ;
127     else
128               dat->pos = TBL_DATA_DATA;
129
130     if (TBL_CELL_HORIZ == dat->layout->pos ||
131         TBL_CELL_DHORIZ == dat->layout->pos ||
132         TBL_CELL_DOWN == dat->layout->pos)
133         if (TBL_DATA_DATA == dat->pos && '\0' != *dat->string)
134             mandoc_msg(MANDOCERR_TBLIGNDATA,
135                       tbl->parse, ln, sv, NULL);
136
137     return(1);
138 }
139
140 /* ARGSUSED */
141 int
142 tbl_cdata(struct tbl_node *tbl, int ln, const char *p)
143 {
144     struct tbl_dat *dat;
145     size_t sz;
146     int pos;
147
148     pos = 0;
149
150     dat = tbl->last_span->last;
151
152     if (p[pos] == 'T' && p[pos + 1] == ')') {
153         pos += 2;
154         if (p[pos] == tbl->opts.tab) {
155             tbl->part = TBL_PART_DATA;
156             pos++;
157             return(data(tbl, tbl->last_span, ln, p, &pos));
158         } else if ('\0' == p[pos]) {
159             tbl->part = TBL_PART_DATA;
160             return(1);
161         }
162     }
163     /* Fallthrough: T} is part of a word. */
164 }
165
166 dat->pos = TBL_DATA_DATA;
167
168 if (dat->string) {
169     sz = strlen(p) + strlen(dat->string) + 2;
170     dat->string = mandoc_realloc(dat->string, sz);
171     strlcat(dat->string, " ", sz);
172     strlcat(dat->string, p, sz);
173 } else
174     dat->string = mandoc_strdup(p);
175
176 if (TBL_CELL_DOWN == dat->layout->pos)
177     mandoc_msg(MANDOCERR_TBLIGNDATA,
178               tbl->parse, ln, pos, NULL);
179
180 return(0);
181 }
182
183 static struct tbl_span *
184 newspan(struct tbl_node *tbl, int line, struct tbl_row *rp)
185 {
186     struct tbl_span *dp;
187
188     dp = mandoc_calloc(1, sizeof(struct tbl_span));

```

```

189     dp->line = line;
190     dp->tbl = &tbl->opts;
191     dp->layout = rp;
192     dp->head = tbl->first_head;
193
194     if (tbl->last_span) {
195         tbl->last_span->next = dp;
196         tbl->last_span = dp;
197     } else {
198         tbl->last_span = tbl->first_span = dp;
199         tbl->current_span = NULL;
200         dp->flags |= TBL_SPAN_FIRST;
201     }
202
203     return(dp);
204 }
205
206 int
207 tbl_data(struct tbl_node *tbl, int ln, const char *p)
208 {
209     struct tbl_span *dp;
210     struct tbl_row *rp;
211     int pos;
212
213     pos = 0;
214
215     if ('\0' == p[pos]) {
216         mandoc_msg(MANDOCERR_TBL, tbl->parse, ln, pos, NULL);
217         return(0);
218     }
219
220     /*
221     * Choose a layout row: take the one following the last parsed
222     * span's. If that doesn't exist, use the last parsed span's.
223     * If there's no last parsed span, use the first row. Lastly,
224     * if the last span was a horizontal line, use the same layout
225     * (it doesn't "consume" the layout).
226     */
227
228     if (tbl->last_span) {
229         assert(tbl->last_span->layout);
230         if (tbl->last_span->pos == TBL_SPAN_DATA) {
231             for (rp = tbl->last_span->layout->next;
232                  rp && rp->first; rp = rp->next) {
233                 switch (rp->first->pos) {
234                     case (TBL_CELL_HORIZ):
235                         dp = newspan(tbl, ln, rp);
236                         dp->pos = TBL_SPAN_HORIZ;
237                         continue;
238                     case (TBL_CELL_DHORIZ):
239                         dp = newspan(tbl, ln, rp);
240                         dp->pos = TBL_SPAN_DHORIZ;
241                         continue;
242                     default:
243                         break;
244                 }
245                 break;
246             }
247         } else
248             rp = tbl->last_span->layout;
249
250         if (NULL == rp)
251             rp = tbl->last_span->layout;
252     } else
253         rp = tbl->first_row;

```

```
255     assert(rp);
257     dp = newspan(tbl, ln, rp);
259     if ( ! strcmp(p, "_") ) {
260         dp->pos = TBL_SPAN_HORIZ;
261         return(1);
262     } else if ( ! strcmp(p, "=") ) {
263         dp->pos = TBL_SPAN_DHORIZ;
264         return(1);
265     }
267     dp->pos = TBL_SPAN_DATA;
269     /* This returns 0 when TBL_PART_CDATA is entered. */
271     while ('\\0' != p[pos])
272         if ( ! data(tbl, dp, ln, p, &pos) )
273             return(0);
275     return(1);
276 }
```

```

*****
3315 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/tbl_html.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: tbl_html.c,v 1.9 2011/09/18 14:14:15 schwarze Exp $ */
2 /*
3  * Copyright (c) 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <assert.h>
22 #include <stdio.h>
23 #include <stdlib.h>
24 #include <string.h>
25
26 #include "mandoc.h"
27 #include "out.h"
28 #include "html.h"
29
30 static void      html_tblopen(struct html *, const struct tbl_span *);
31 static size_t   html_tbl_len(size_t, void *);
32 static size_t   html_tbl_strlen(const char *, void *);
33
34 /* ARGSUSED */
35 static size_t
36 html_tbl_len(size_t sz, void *arg)
37 {
38
39     return(sz);
40 }
41
42 /* ARGSUSED */
43 static size_t
44 html_tbl_strlen(const char *p, void *arg)
45 {
46
47     return(strlen(p));
48 }
49
50 static void
51 html_tblopen(struct html *h, const struct tbl_span *sp)
52 {
53     const struct tbl_head *hp;
54     struct htmlpair tag;
55     struct roffsu    su;
56     struct roffcol   *col;

```

```

58     if (TBL_SPAN_FIRST & sp->flags) {
59         h->tbl.len = html_tbl_len;
60         h->tbl.slen = html_tbl_strlen;
61         tblcalc(&h->tbl, sp);
62     }
63
64     assert(NULL == h->tblt);
65     PAIR_CLASS_INIT(&tag, "tbl");
66     h->tblt = print_otag(h, TAG_TABLE, 1, &tag);
67
68     for (hp = sp->head; hp; hp = hp->next) {
69         bufinit(h);
70         col = &h->tbl.cols[hp->ident];
71         SCALE_HS_INIT(&su, col->width);
72         bufcat_su(h, "width", &su);
73         PAIR_STYLE_INIT(&tag, h);
74         print_otag(h, TAG_COL, 1, &tag);
75     }
76
77     print_otag(h, TAG_TBODY, 0, NULL);
78 }
79
80 void
81 print_tblclose(struct html *h)
82 {
83
84     assert(h->tblt);
85     print_tagq(h, h->tblt);
86     h->tblt = NULL;
87 }
88
89 void
90 print_tbl(struct html *h, const struct tbl_span *sp)
91 {
92     const struct tbl_head *hp;
93     const struct tbl_dat *dp;
94     struct htmlpair tag;
95     struct tag          *tt;
96
97     /* Inhibit printing of spaces: we do padding ourselves. */
98
99     if (NULL == h->tblt)
100         html_tblopen(h, sp);
101
102     assert(h->tblt);
103
104     h->flags |= HTML_NONOSPACE;
105     h->flags |= HTML_NOSPACE;
106
107     tt = print_otag(h, TAG_TR, 0, NULL);
108
109     switch (sp->pos) {
110     case (TBL_SPAN_HORIZ):
111         /* FALLTHROUGH */
112     case (TBL_SPAN_DHORIZ):
113         PAIR_INIT(&tag, ATTR_COLSPAN, "0");
114         print_otag(h, TAG_TD, 1, &tag);
115         break;
116     default:
117         dp = sp->first;
118         for (hp = sp->head; hp; hp = hp->next) {
119             print_stagq(h, tt);
120             print_otag(h, TAG_TD, 0, NULL);
121         }
122         switch (hp->pos) {

```

```
123         case (TBL_HEAD_VERT):
124             /* FALLTHROUGH */
125         case (TBL_HEAD_DVERT):
126             continue;
127         case (TBL_HEAD_DATA):
128             if (NULL == dp)
129                 break;
130             if (TBL_CELL_DOWN != dp->layout->pos)
131                 if (dp->string)
132                     print_text(h, dp->string);
133             dp = dp->next;
134             break;
135     }
136     break;
137 }
138
139 print_tagq(h, tt);
140
141 h->flags &= ~HTML_NONOSPACE;
142
143 if (TBL_SPAN_LAST & sp->flags) {
144     assert(h->tbl.cols);
145     free(h->tbl.cols);
146     h->tbl.cols = NULL;
147     print_tblclose(h);
148 }
149
150 }
151 }
```

```

*****
10207 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/tbl_layout.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: tbl_layout.c,v 1.22 2011/09/18 14:14:15 schwarze Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <assert.h>
22 #include <ctype.h>
23 #include <stdlib.h>
24 #include <string.h>
25 #include <time.h>
26
27 #include "mandoc.h"
28 #include "libmandoc.h"
29 #include "libroff.h"
30
31 struct tbl_phrase {
32     char      name;
33     enum tbl_cellt key;
34 };
35
36 /*
37  * FIXME: we can make this parse a lot nicer by, when an error is
38  * encountered in a layout key, bailing to the next key (i.e. to the
39  * next whitespace then continuing).
40 */
41
42 #define KEYS_MAX      11
43
44 static const struct tbl_phrase keys[KEYS_MAX] = {
45     { 'c',      TBL_CELL_CENTRE },
46     { 'r',      TBL_CELL_RIGHT },
47     { 'l',      TBL_CELL_LEFT },
48     { 'n',      TBL_CELL_NUMBER },
49     { 's',      TBL_CELL_SPAN },
50     { 'a',      TBL_CELL_LONG },
51     { '^',      TBL_CELL_DOWN },
52     { '-',      TBL_CELL_HORIZ },
53     { '|',      TBL_CELL_HORIZ },
54     { '=',      TBL_CELL_DHORIZ },
55     { '|',      TBL_CELL_VERT },
56 };

```

```

58 static int      mods(struct tbl_node *, struct tbl_cell *,
59                    int, const char *, int *);
60 static int      cell(struct tbl_node *, struct tbl_row *,
61                    int, const char *, int *);
62 static void      row(struct tbl_node *, int, const char *, int *);
63 static struct tbl_cell *cell_alloc(struct tbl_node *,
64                                   struct tbl_row *, enum tbl_cellt);
65 static void      head_adjust(const struct tbl_cell *,
66                             struct tbl_head *);
67
68 static int
69 mods(struct tbl_node *tbl, struct tbl_cell *cp,
70       int ln, const char *p, int *pos)
71 {
72     char      buf[5];
73     int       i;
74
75     /* Not all types accept modifiers. */
76
77     switch (cp->pos) {
78     case (TBL_CELL_DOWN):
79         /* FALLTHROUGH */
80     case (TBL_CELL_HORIZ):
81         /* FALLTHROUGH */
82     case (TBL_CELL_DHORIZ):
83         /* FALLTHROUGH */
84     case (TBL_CELL_VERT):
85         /* FALLTHROUGH */
86     case (TBL_CELL_DVERT):
87         return(1);
88     default:
89         break;
90     }
91
92 mod:
93     /*
94      * XXX: since, at least for now, modifiers are non-conflicting
95      * (are separable by value, regardless of position), we let
96      * modifiers come in any order. The existing tbl doesn't let
97      * this happen.
98      */
99     switch (p[*pos]) {
100    case ('\0'):
101        /* FALLTHROUGH */
102    case (' '):
103        /* FALLTHROUGH */
104    case ('\t'):
105        /* FALLTHROUGH */
106    case (','):
107        /* FALLTHROUGH */
108    case ('.'):
109        return(1);
110    default:
111        break;
112    }
113
114     /* Throw away parenthesised expression. */
115
116     if (('(' == p[*pos]) {
117         (*pos)++;
118         while (p[*pos] && ')' != p[*pos])
119             (*pos)++;
120         if (')' == p[*pos]) {
121             (*pos)++;
122             goto mod;

```

```

123     }
124     mandoc_msg(MANDOCERR_TBLAYOUT,
125               tbl->parse, ln, *pos, NULL);
126     return(0);
127 }
128
129 /* Parse numerical spacing from modifier string. */
130
131 if (isdigit((unsigned char)p[*pos])) {
132     for (i = 0; i < 4; i++) {
133         if (! isdigit((unsigned char)p[*pos + i]))
134             break;
135         buf[i] = p[*pos + i];
136     }
137     buf[i] = '\0';
138
139     /* No greater than 4 digits. */
140
141     if (4 == i) {
142         mandoc_msg(MANDOCERR_TBLAYOUT, tbl->parse,
143                   ln, *pos, NULL);
144         return(0);
145     }
146
147     *pos += i;
148     cp->spacing = (size_t)atoi(buf);
149
150     goto mod;
151     /* NOTREACHED */
152 }
153
154 /* TODO: GNU has many more extensions. */
155
156 switch (tolower((unsigned char)p[*pos])) {
157 case ('z'):
158     cp->flags |= TBL_CELL_WIGN;
159     goto mod;
160 case ('u'):
161     cp->flags |= TBL_CELL_UP;
162     goto mod;
163 case ('e'):
164     cp->flags |= TBL_CELL_EQUAL;
165     goto mod;
166 case ('t'):
167     cp->flags |= TBL_CELL_TALIGN;
168     goto mod;
169 case ('d'):
170     cp->flags |= TBL_CELL_BALIGN;
171     goto mod;
172 case ('w'): /* XXX for now, ignore minimal column width */
173     goto mod;
174 case ('f'):
175     break;
176 case ('r'):
177     /* FALLTHROUGH */
178 case ('b'):
179     /* FALLTHROUGH */
180 case ('i'):
181     (*pos)--;
182     break;
183 default:
184     mandoc_msg(MANDOCERR_TBLAYOUT, tbl->parse,
185               ln, *pos - 1, NULL);
186     return(0);
187 }

```

```

189     switch (tolower((unsigned char)p[*pos])) {
190     case ('3'):
191         /* FALLTHROUGH */
192     case ('b'):
193         cp->flags |= TBL_CELL_BOLD;
194         goto mod;
195     case ('2'):
196         /* FALLTHROUGH */
197     case ('i'):
198         cp->flags |= TBL_CELL_ITALIC;
199         goto mod;
200     case ('1'):
201         /* FALLTHROUGH */
202     case ('r'):
203         goto mod;
204     default:
205         break;
206     }
207
208     mandoc_msg(MANDOCERR_TBLAYOUT, tbl->parse,
209               ln, *pos - 1, NULL);
210     return(0);
211 }
212
213 static int
214 cell(struct tbl_node *tbl, struct tbl_row *rp,
215       int ln, const char *p, int *pos)
216 {
217     int i;
218     enum tbl_cellt c;
219
220     /* Parse the column position ('r', 'R', '|', ...). */
221
222     for (i = 0; i < KEYS_MAX; i++)
223         if (tolower((unsigned char)p[*pos]) == keys[i].name)
224             break;
225
226     if (KEYS_MAX == i) {
227         mandoc_msg(MANDOCERR_TBLAYOUT, tbl->parse,
228                   ln, *pos, NULL);
229         return(0);
230     }
231
232     c = keys[i].key;
233
234     /*
235      * If a span cell is found first, raise a warning and abort the
236      * parse. If a span cell is found and the last layout element
237      * isn't a "normal" layout, bail.
238      *
239      * FIXME: recover from this somehow
240      */
241
242     if (TBL_CELL_SPAN == c) {
243         if (NULL == rp->first) {
244             mandoc_msg(MANDOCERR_TBLAYOUT, tbl->parse,
245                       ln, *pos, NULL);
246             return(0);
247         } else if (rp->last)
248             switch (rp->last->pos) {
249             case (TBL_CELL_VERT):
250             case (TBL_CELL_DVERT):
251             case (TBL_CELL_HORIZ):
252             case (TBL_CELL_DHORIZ):
253                 mandoc_msg(MANDOCERR_TBLAYOUT, tbl->parse,
254                           ln, *pos, NULL);

```

```

255         return(0);
256     default:
257         break;
258     }
259 }

261 /*
262  * If a vertical spanner is found, we may not be in the first
263  * row.
264  */

266 if (TBL_CELL_DOWN == c && rp == tbl->first_row) {
267     mandoc_msg(MANDOCERR_TBLAYOUT, tbl->parse, ln, *pos, NULL);
268     return(0);
269 }

271 (*pos)++;

273 /* Extra check for the double-vertical. */

275 if (TBL_CELL_VERT == c && '|' == p[*pos]) {
276     (*pos)++;
277     c = TBL_CELL_DVERT;
278 }

279 /* Disallow adjacent spacers. */

282 if (rp->last && (TBL_CELL_VERT == c || TBL_CELL_DVERT == c) &&
283     (TBL_CELL_VERT == rp->last->pos ||
284     TBL_CELL_DVERT == rp->last->pos)) {
285     mandoc_msg(MANDOCERR_TBLAYOUT, tbl->parse, ln, *pos - 1, NULL);
286     return(0);
287 }

289 /* Allocate cell then parse its modifiers. */

291 return(mods(tbl, cell_alloc(tbl, rp, c), ln, p, pos));
292 }

295 static void
296 row(struct tbl_node *tbl, int ln, const char *p, int *pos)
297 {
298     struct tbl_row *rp;

300 row: /*
301     * EBNF describing this section:
302     *
303     * row      ::= row_list [:space:]* [.]?[\n]
304     * row_list ::= [:space:]* row_elem row_tail
305     * row_tail ::= [:space:]*[, ] row_list |
306     *             epsilon
307     * row_elem ::= [\t\ ]*[:alpha:]+
308     */

310     rp = mandoc_calloc(1, sizeof(struct tbl_row));
311     if (tbl->last_row) {
312         tbl->last_row->next = rp;
313         tbl->last_row = rp;
314     } else
315         tbl->last_row = tbl->first_row = rp;

317 cell:
318     while (isspace((unsigned char)p[*pos]))
319         (*pos)++;

```

```

321     /* Safely exit layout context. */

323     if ('.' == p[*pos]) {
324         tbl->part = TBL_PART_DATA;
325         if (NULL == tbl->first_row)
326             mandoc_msg(MANDOCERR_TBLNOLAYOUT, tbl->parse,
327                 ln, *pos, NULL);
328         (*pos)++;
329         return;
330     }

332     /* End (and possibly restart) a row. */

334     if (',' == p[*pos]) {
335         (*pos)++;
336         goto row;
337     } else if ('\0' == p[*pos])
338         return;

340     if (! cell(tbl, rp, ln, p, pos))
341         return;

343     goto cell;
344     /* NOTREACHED */
345 }

347 int
348 tbl_layout(struct tbl_node *tbl, int ln, const char *p)
349 {
350     int pos;

352     pos = 0;
353     row(tbl, ln, p, &pos);

355     /* Always succeed. */
356     return(1);
357 }

359 static struct tbl_cell *
360 cell_alloc(struct tbl_node *tbl, struct tbl_row *rp, enum tbl_cellt pos)
361 {
362     struct tbl_cell *p, *pp;
363     struct tbl_head *h, *hp;

365     p = mandoc_calloc(1, sizeof(struct tbl_cell));

367     if (NULL != (pp = rp->last)) {
368         rp->last->next = p;
369         rp->last = p;
370     } else
371         rp->last = rp->first = p;

373     p->pos = pos;

375     /*
376     * This is a little bit complicated. Here we determine the
377     * header the corresponds to a cell. We add headers dynamically
378     * when need be or re-use them, otherwise. As an example, given
379     * the following:
380     *
381     *     1  c  | |  1
382     *     2  |  c  |  1
383     *     3  1  1
384     *     3  | |  c  |  1  |.
385     *
386     * We first add the new headers (as there are none) in (1); then

```

```

387  * in (2) we insert the first spanner (as it doesn't match up
388  * with the header); then we re-use the prior data headers,
389  * skipping over the spanners; then we re-use everything and add
390  * a last spanner. Note that VERT headers are made into DVERT
391  * ones.
392  */
394  h = pp ? pp->head->next : tbl->first_head;

396  if (h) {
397      /* Re-use data header. */
398      if (TBL_HEAD_DATA == h->pos &&
399          (TBL_CELL_VERT != p->pos &&
400           TBL_CELL_DVERT != p->pos)) {
401          p->head = h;
402          return(p);
403      }

405      /* Re-use spanner header. */
406      if (TBL_HEAD_DATA != h->pos &&
407          (TBL_CELL_VERT == p->pos ||
408           TBL_CELL_DVERT == p->pos)) {
409          head_adjust(p, h);
410          p->head = h;
411          return(p);
412      }

414      /* Right-shift headers with a new spanner. */
415      if (TBL_HEAD_DATA == h->pos &&
416          (TBL_CELL_VERT == p->pos ||
417           TBL_CELL_DVERT == p->pos)) {
418          hp = mandoc_calloc(1, sizeof(struct tbl_head));
419          hp->ident = tbl->opts.cols++;
420          hp->prev = h->prev;
421          if (h->prev)
422              h->prev->next = hp;
423          if (h == tbl->first_head)
424              tbl->first_head = hp;
425          h->prev = hp;
426          hp->next = h;
427          head_adjust(p, hp);
428          p->head = hp;
429          return(p);
430      }

432      if (NULL != (h = h->next)) {
433          head_adjust(p, h);
434          p->head = h;
435          return(p);
436      }

438      /* Fall through to default case... */
439  }

441  hp = mandoc_calloc(1, sizeof(struct tbl_head));
442  hp->ident = tbl->opts.cols++;

444  if (tbl->last_head) {
445      hp->prev = tbl->last_head;
446      tbl->last_head->next = hp;
447      tbl->last_head = hp;
448  } else
449      tbl->last_head = tbl->first_head = hp;

451  head_adjust(p, hp);
452  p->head = hp;

```

```

453      return(p);
454  }

456  static void
457  head_adjust(const struct tbl_cell *cellp, struct tbl_head *head)
458  {
459      if (TBL_CELL_VERT != cellp->pos &&
460          TBL_CELL_DVERT != cellp->pos) {
461          head->pos = TBL_HEAD_DATA;
462          return;
463      }

465      if (TBL_CELL_VERT == cellp->pos)
466          if (TBL_HEAD_DVERT != head->pos)
467              head->pos = TBL_HEAD_VERT;

469      if (TBL_CELL_DVERT == cellp->pos)
470          head->pos = TBL_HEAD_DVERT;
471  }

```



```

*****
6142 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/tbl_opts.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: tbl_opts.c,v 1.12 2011/09/18 14:14:15 schwarze Exp $ */
2 /*
3  * Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <ctype.h>
22 #include <stdio.h>
23 #include <stdlib.h>
24 #include <string.h>
25
26 #include "mandoc.h"
27 #include "libmandoc.h"
28 #include "libroff.h"
29
30 enum tbl_ident {
31     KEY_CENTRE = 0,
32     KEY_DELIM,
33     KEY_EXPAND,
34     KEY_BOX,
35     KEY_DBOX,
36     KEY_ALLBOX,
37     KEY_TAB,
38     KEY_LINESIZE,
39     KEY_NOKEEP,
40     KEY_DPOINT,
41     KEY_NOSPACE,
42     KEY_FRAME,
43     KEY_DFRAME,
44     KEY_MAX
45 };
46
47 struct tbl_phrase {
48     const char *name;
49     int key;
50     enum tbl_ident ident;
51 };
52
53 /* Handle Commonwealth/American spellings. */
54 #define KEY_MAXKEYS 14
55
56 /* Maximum length of key name string. */

```

```

57 #define KEY_MAXNAME 13
58
59 /* Maximum length of key number size. */
60 #define KEY_MAXNUMSZ 10
61
62 static const struct tbl_phrase keys[KEY_MAXKEYS] = {
63     {"center", TBL_OPT_CENTRE, KEY_CENTRE},
64     {"centre", TBL_OPT_CENTRE, KEY_CENTRE},
65     {"delim", 0, KEY_DELIM},
66     {"expand", TBL_OPT_EXPAND, KEY_EXPAND},
67     {"box", TBL_OPT_BOX, KEY_BOX},
68     {"doublebox", TBL_OPT_DBOX, KEY_DBOX},
69     {"allbox", TBL_OPT_ALLBOX, KEY_ALLBOX},
70     {"frame", TBL_OPT_BOX, KEY_FRAME},
71     {"doubleframe", TBL_OPT_DBOX, KEY_DFRAME},
72     {"tab", 0, KEY_TAB},
73     {"linesize", 0, KEY_LINESIZE},
74     {"nokeep", TBL_OPT_NOKEEP, KEY_NOKEEP},
75     {"decimalpoint", 0, KEY_DPOINT},
76     {"nospaces", TBL_OPT_NOSPACE, KEY_NOSPACE},
77 };
78
79 static int
80 arg(struct tbl_node *, int, const char *, int *, enum tbl_ident);
81 static void
82 opt(struct tbl_node *, int, const char *, int *);
83
84 static int
85 arg(struct tbl_node *tbl, int ln, const char *p, int *pos, enum tbl_ident key)
86 {
87     int i;
88     char buf[KEY_MAXNUMSZ];
89
90     while (isspace((unsigned char)p[*pos]))
91         (*pos)++;
92
93     /* Arguments always begin with a parenthesis. */
94
95     if (('!' != p[*pos]) {
96         mandoc_msg(MANDOCERR_TBL, tbl->parse,
97                 ln, *pos, NULL);
98         return(0);
99     }
100
101     (*pos)++;
102
103     /*
104      * The arguments can be ANY value, so we can't just stop at the
105      * next close parenthesis (the argument can be a closed
106      * parenthesis itself).
107      */
108
109     switch (key) {
110     case (KEY_DELIM):
111         if ('\0' == p[(*pos)+1]) {
112             mandoc_msg(MANDOCERR_TBL, tbl->parse,
113                     ln, *pos - 1, NULL);
114             return(0);
115         }
116
117         if ('\0' == p[(*pos)+2]) {
118             mandoc_msg(MANDOCERR_TBL, tbl->parse,
119                     ln, *pos - 1, NULL);
120             return(0);
121         }
122         break;

```

```

123     case (KEY_TAB):
124         if ('\0' != (tbl->opts.tab = p[(*pos)+]))
125             break;
127         mandoc_msg(MANDOCERR_TBL, tbl->parse,
128                 ln, *pos - 1, NULL);
129         return(0);
130     case (KEY_LINESIZE):
131         for (i = 0; i < KEY_MAXNUMSZ && p[*pos]; i++, (*pos)++) {
132             buf[i] = p[*pos];
133             if (! isdigit((unsigned char)buf[i]))
134                 break;
135         }
137         if (i < KEY_MAXNUMSZ) {
138             buf[i] = '\0';
139             tbl->opts.linesize = atoi(buf);
140             break;
141         }
143         mandoc_msg(MANDOCERR_TBL, tbl->parse, ln, *pos, NULL);
144         return(0);
145     case (KEY_DPOINT):
146         if ('\0' != (tbl->opts.decimal = p[(*pos)+]))
147             break;
149         mandoc_msg(MANDOCERR_TBL, tbl->parse,
150                 ln, *pos - 1, NULL);
151         return(0);
152     default:
153         abort();
154         /* NOTREACHED */
155 }
157 /* End with a close parenthesis. */
159 if (')' == p[(*pos)+])
160     return(1);
162 mandoc_msg(MANDOCERR_TBL, tbl->parse, ln, *pos - 1, NULL);
163 return(0);
164 }
166 static void
167 opt(struct tbl_node *tbl, int ln, const char *p, int *pos)
168 {
169     int         i, sv;
170     char        buf[KEY_MAXNAME];
172     /*
173      * Parse individual options from the stream as surrounded by
174      * this goto.  Each pass through the routine parses out a single
175      * option and registers it.  Option arguments are processed in
176      * the arg() function.
177      */
179     again: /*
180          * EBNF describing this section:
181          *
182          * options      ::= option_list [:space:]* [;][\n]
183          * option_list ::= option option_tail
184          * option_tail ::= [:space:]+ option_list |
185          *                  ::= epsilon
186          * option      ::= [:alpha:]+ args
187          * args        ::= [:space:]* [(:[:alpha:]+)]
188          */

```

```

190     while (isspace((unsigned char)p[*pos]))
191         (*pos)++;
193     /* Safe exit point. */
195     if (',' == p[*pos])
196         return;
198     /* Copy up to first non-alpha character. */
200     for (sv = *pos, i = 0; i < KEY_MAXNAME; i++, (*pos)++) {
201         buf[i] = (char)tolower((unsigned char)p[*pos]);
202         if (! isalpha((unsigned char)buf[i]))
203             break;
204     }
206     /* Exit if buffer is empty (or overrun). */
208     if (KEY_MAXNAME == i || 0 == i) {
209         mandoc_msg(MANDOCERR_TBL, tbl->parse, ln, *pos, NULL);
210         return;
211     }
213     buf[i] = '\0';
215     while (isspace((unsigned char)p[*pos]))
216         (*pos)++;
218     /*
219      * Look through all of the available keys to find one that
220      * matches the input.  FIXME: hashtable this.
221      */
223     for (i = 0; i < KEY_MAXKEYS; i++) {
224         if (strcmp(buf, keys[i].name))
225             continue;
227         /*
228          * Note: this is more difficult to recover from, as we
229          * can be anywhere in the option sequence and it's
230          * harder to jump to the next.  Meanwhile, just bail out
231          * of the sequence altogether.
232          */
234         if (keys[i].key)
235             tbl->opts.opts |= keys[i].key;
236         else if (! arg(tbl, ln, p, pos, keys[i].ident))
237             return;
239         break;
240     }
242     /*
243      * Allow us to recover from bad options by continuing to another
244      * parse sequence.
245      */
247     if (KEY_MAXKEYS == i)
248         mandoc_msg(MANDOCERR_TBL, tbl->parse, ln, sv, NULL);
250     goto again;
251     /* NOTREACHED */
252 }
254 int

```

```
255 tbl_option(struct tbl_node *tbl, int ln, const char *p)
256 {
257     int          pos;
258
259     /*
260      * Table options are always on just one line, so automatically
261      * switch into the next input mode here.
262      */
263     tbl->part = TBL_PART_LAYOUT;
264
265     pos = 0;
266     opt(tbl, ln, p, &pos);
267
268     /* Always succeed. */
269     return(1);
270 }
```

```

*****
10029 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/tbl_term.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: tbl_term.c,v 1.21 2011/09/20 23:05:49 schwarze Exp $ */
2 /*
3  * Copyright (c) 2009, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <assert.h>
23 #include <stdio.h>
24 #include <stdlib.h>
25 #include <string.h>
26
27 #include "mandoc.h"
28 #include "out.h"
29 #include "term.h"
30
31 static size_t term_tbl_len(size_t, void *);
32 static size_t term_tbl_strlen(const char *, void *);
33 static void tbl_char(struct term *, char, size_t);
34 static void tbl_data(struct term *, const struct tbl *,
35                     const struct tbl_dat *,
36                     const struct roffcol *);
37 static size_t tbl_rulewidth(struct term *, const struct tbl_head *);
38 static void tbl_hframe(struct term *, const struct tbl_span *, int);
39 static void tbl_literal(struct term *, const struct tbl_dat *,
40                        const struct roffcol *);
41 static void tbl_number(struct term *, const struct tbl *,
42                       const struct tbl_dat *,
43                       const struct roffcol *);
44 static void tbl_hrule(struct term *, const struct tbl_span *);
45 static void tbl_vrule(struct term *, const struct tbl_head *);
46
47 static size_t
48 term_tbl_strlen(const char *p, void *arg)
49 {
50     return(term_strlen((const struct term *)arg, p));
51 }
52
53
54 static size_t
55 term_tbl_len(size_t sz, void *arg)

```

```

57 {
58     return(term_len((const struct term *)arg, sz));
59 }
60
61 void
62 term_tbl(struct term *tp, const struct tbl_span *sp)
63 {
64     const struct tbl_head *hp;
65     const struct tbl_dat *dp;
66     struct roffcol *col;
67     int spans;
68     size_t rmargin, maxrmargin;
69
70     rmargin = tp->rmargin;
71     maxrmargin = tp->maxrmargin;
72
73     tp->rmargin = tp->maxrmargin = TERM_MAXMARGIN;
74
75     /* Inhibit printing of spaces: we do padding ourselves. */
76
77     tp->flags |= TERM_NONOSPACE;
78     tp->flags |= TERMP_NOSPACE;
79
80     /*
81      * The first time we're invoked for a given table block,
82      * calculate the table widths and decimal positions.
83      */
84
85     if (TBL_SPAN_FIRST & sp->flags) {
86         term_flushln(tp);
87
88         tp->tbl.len = term_tbl_len;
89         tp->tbl.slen = term_tbl_strlen;
90         tp->tbl.arg = tp;
91
92         tblcalc(&tp->tbl, sp);
93     }
94
95     /* Horizontal frame at the start of boxed tables. */
96
97     if (TBL_SPAN_FIRST & sp->flags) {
98         if (TBL_OPT_DBOX & sp->tbl->opts)
99             tbl_hframe(tp, sp, 1);
100         if (TBL_OPT_DBOX & sp->tbl->opts ||
101             TBL_OPT_BOX & sp->tbl->opts)
102             tbl_hframe(tp, sp, 0);
103     }
104
105     /* Vertical frame at the start of each row. */
106
107     if (TBL_OPT_BOX & sp->tbl->opts || TBL_OPT_DBOX & sp->tbl->opts)
108         term_word(tp, TBL_SPAN_HORIZ == sp->pos ||
109                 TBL_SPAN_DHORIZ == sp->pos ? "+" : "|");
110
111     /*
112      * Now print the actual data itself depending on the span type.
113      * Spanner spans get a horizontal rule; data spanners have their
114      * data printed by matching data to header.
115      */
116
117     switch (sp->pos) {
118     case (TBL_SPAN_HORIZ):
119         /* FALLTHROUGH */
120     case (TBL_SPAN_DHORIZ):
121         tbl_hrule(tp, sp);
122     }

```

```

123     break;
124     case (TBL_SPAN_DATA):
125         /* Iterate over template headers. */
126         dp = sp->first;
127         spans = 0;
128         for (hp = sp->head; hp; hp = hp->next) {
129             /*
130              * If the current data header is invoked during
131              * a spanner ("spans" > 0), don't emit anything
132              * at all.
133              */
134             switch (hp->pos) {
135                 case (TBL_HEAD_VERT):
136                     /* FALLTHROUGH */
137                 case (TBL_HEAD_DVERT):
138                     if (spans <= 0)
139                         tbl_vrule(tp, hp);
140                     continue;
141                 case (TBL_HEAD_DATA):
142                     break;
143             }
144
145             if (--spans >= 0)
146                 continue;
147
148             /*
149              * All cells get a leading blank, except the
150              * first one and those after double rulers.
151              */
152
153             if (hp->prev && TBL_HEAD_DVERT != hp->prev->pos)
154                 tbl_char(tp, ASCII_NBRSP, 1);
155
156             col = &tp->tbl.cols[hp->ident];
157             tbl_data(tp, sp->tbl, dp, col);
158
159             /* No trailing blanks. */
160
161             if (NULL == hp->next)
162                 break;
163
164             /*
165              * Add another blank between cells,
166              * or two when there is no vertical ruler.
167              */
168
169             tbl_char(tp, ASCII_NBRSP,
170                 TBL_HEAD_VERT == hp->next->pos ||
171                 TBL_HEAD_DVERT == hp->next->pos ? 1 : 2);
172
173             /*
174              * Go to the next data cell and assign the
175              * number of subsequent spans, if applicable.
176              */
177
178             if (dp) {
179                 spans = dp->spans;
180                 dp = dp->next;
181             }
182         }
183     }
184 }
185
186 /* Vertical frame at the end of each row. */
187
188 if (TBL_OPT_BOX & sp->tbl->opts || TBL_OPT_DBOX & sp->tbl->opts)

```

```

189     term_word(tp, TBL_SPAN_HORIZ == sp->pos ||
190         TBL_SPAN_DHORIZ == sp->pos ? "+" : " |");
191     term_flushln(tp);
192
193     /*
194      * If we're the last row, clean up after ourselves: clear the
195      * existing table configuration and set it to NULL.
196      */
197
198     if (TBL_SPAN_LAST & sp->flags) {
199         if (TBL_OPT_DBOX & sp->tbl->opts ||
200             TBL_OPT_BOX & sp->tbl->opts)
201             tbl_hframe(tp, sp, 0);
202         if (TBL_OPT_DBOX & sp->tbl->opts)
203             tbl_hframe(tp, sp, 1);
204         assert(tp->tbl.cols);
205         free(tp->tbl.cols);
206         tp->tbl.cols = NULL;
207     }
208
209     tp->flags &= ~TERMP_NONOSPACE;
210     tp->rmargin = rmargin;
211     tp->maxrmargin = maxrmargin;
212 }
213
214 /*
215  * Horizontal rules extend across the entire table.
216  * Calculate the width by iterating over columns.
217  */
218 static size_t
219 tbl_rulewidth(struct term *tp, const struct tbl_head *hp)
220 {
221     size_t      width;
222
223     width = tp->tbl.cols[hp->ident].width;
224     if (TBL_HEAD_DATA == hp->pos) {
225         /* Account for leading blanks. */
226         if (hp->prev && TBL_HEAD_DVERT != hp->prev->pos)
227             width++;
228         /* Account for trailing blanks. */
229         width++;
230         if (hp->next &&
231             TBL_HEAD_VERT != hp->next->pos &&
232             TBL_HEAD_DVERT != hp->next->pos)
233             width++;
234     }
235     return(width);
236 }
237
238 /*
239  * Rules inside the table can be single or double
240  * and have crossings with vertical rules marked with pluses.
241  */
242 static void
243 tbl_hrule(struct term *tp, const struct tbl_span *sp)
244 {
245     const struct tbl_head *hp;
246     char      c;
247
248     c = '-';
249     if (TBL_SPAN_DHORIZ == sp->pos)
250         c = '=';
251
252     for (hp = sp->head; hp; hp = hp->next)
253         tbl_char(tp,

```

```

255         TBL_HEAD_DATA == hp->pos ? c : '+',
256         tbl_rulewidth(tp, hp));
257 }

259 /*
260  * Rules above and below the table are always single
261  * and have an additional plus at the beginning and end.
262  * For double frames, this function is called twice,
263  * and the outer one does not have crossings.
264  */
265 static void
266 tbl_hframe(struct term *tp, const struct tbl_span *sp, int outer)
267 {
268     const struct tbl_head *hp;

270     term_word(tp, "+");
271     for (hp = sp->head; hp; hp = hp->next)
272         tbl_char(tp,
273                 outer || TBL_HEAD_DATA == hp->pos ? '-' : '+',
274                 tbl_rulewidth(tp, hp));
275     term_word(tp, "+");
276     term_flushln(tp);
277 }

279 static void
280 tbl_data(struct term *tp, const struct tbl *tbl,
281          const struct tbl_dat *dp,
282          const struct roffcol *col)
283 {
285     if (NULL == dp) {
286         tbl_char(tp, ASCII_NBRSP, col->width);
287         return;
288     }
289     assert(dp->layout);

291     switch (dp->pos) {
292     case (TBL_DATA_NONE):
293         tbl_char(tp, ASCII_NBRSP, col->width);
294         return;
295     case (TBL_DATA_HORIZ):
296         /* FALLTHROUGH */
297     case (TBL_DATA_NHORIZ):
298         tbl_char(tp, '-', col->width);
299         return;
300     case (TBL_DATA_NDHORIZ):
301         /* FALLTHROUGH */
302     case (TBL_DATA_DHORIZ):
303         tbl_char(tp, '=', col->width);
304         return;
305     default:
306         break;
307     }

309     switch (dp->layout->pos) {
310     case (TBL_CELL_HORIZ):
311         tbl_char(tp, '-', col->width);
312         break;
313     case (TBL_CELL_DHORIZ):
314         tbl_char(tp, '=', col->width);
315         break;
316     case (TBL_CELL_LONG):
317         /* FALLTHROUGH */
318     case (TBL_CELL_CENTRE):
319         /* FALLTHROUGH */
320     case (TBL_CELL_LEFT):

```

```

321         /* FALLTHROUGH */
322     case (TBL_CELL_RIGHT):
323         tbl_literal(tp, dp, col);
324         break;
325     case (TBL_CELL_NUMBER):
326         tbl_number(tp, tbl, dp, col);
327         break;
328     case (TBL_CELL_DOWN):
329         tbl_char(tp, ASCII_NBRSP, col->width);
330         break;
331     default:
332         abort();
333         /* NOTREACHED */
334     }
335 }

337 static void
338 tbl_vrule(struct term *tp, const struct tbl_head *hp)
339 {
341     switch (hp->pos) {
342     case (TBL_HEAD_VERT):
343         term_word(tp, "|");
344         break;
345     case (TBL_HEAD_DVERT):
346         term_word(tp, "||");
347         break;
348     default:
349         break;
350     }
351 }

353 static void
354 tbl_char(struct term *tp, char c, size_t len)
355 {
356     size_t      i, sz;
357     char        cp[2];

359     cp[0] = c;
360     cp[1] = '\0';

362     sz = term_strlen(tp, cp);

364     for (i = 0; i < len; i += sz)
365         term_word(tp, cp);
366 }

368 static void
369 tbl_literal(struct term *tp, const struct tbl_dat *dp,
370            const struct roffcol *col)
371 {
372     size_t      len, padl, padr;

374     assert(dp->string);
375     len = term_strlen(tp, dp->string);
376     padr = col->width > len ? col->width - len : 0;
377     padl = 0;

379     switch (dp->layout->pos) {
380     case (TBL_CELL_LONG):
381         padl = term_len(tp, 1);
382         padr = padr > padl ? padr - padl : 0;
383         break;
384     case (TBL_CELL_CENTRE):
385         if (2 > padr)
386             break;

```

```
387         padl = padr / 2;
388         padr -= padl;
389         break;
390     case (TBL_CELL_RIGHT):
391         padl = padr;
392         padr = 0;
393         break;
394     default:
395         break;
396     }
397
398     tbl_char(tp, ASCII_NBRSP, padl);
399     term_word(tp, dp->string);
400     tbl_char(tp, ASCII_NBRSP, padr);
401 }
402
403 static void
404 tbl_number(struct term *tp, const struct tbl *tbl,
405            const struct tbl_dat *dp,
406            const struct roffcol *col)
407 {
408     char          *cp;
409     char          buf[2];
410     size_t        sz, psz, ssz, d, padl;
411     int           i;
412
413     /*
414      * See calc_data_number(). Left-pad by taking the offset of our
415      * and the maximum decimal; right-pad by the remaining amount.
416      */
417
418     assert(dp->string);
419
420     sz = term_strlen(tp, dp->string);
421
422     buf[0] = tbl->decimal;
423     buf[1] = '\0';
424
425     psz = term_strlen(tp, buf);
426
427     if (NULL != (cp = strrchr(dp->string, tbl->decimal))) {
428         buf[1] = '\0';
429         for (ssz = 0, i = 0; cp != &dp->string[i]; i++) {
430             buf[0] = dp->string[i];
431             ssz += term_strlen(tp, buf);
432         }
433         d = ssz + psz;
434     } else
435         d = sz + psz;
436
437     padl = col->decimal - d;
438
439     tbl_char(tp, ASCII_NBRSP, padl);
440     term_word(tp, dp->string);
441     if (col->width > sz + padl)
442         tbl_char(tp, ASCII_NBRSP, col->width - sz - padl);
443 }
```

```

*****
15531 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/term.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /*      $Id: term.c,v 1.201 2011/09/21 09:57:13 schwarze Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  * Copyright (c) 2010, 2011 Ingo Schwarze <schwarze@openbsd.org>
5  *
6  * Permission to use, copy, modify, and distribute this software for any
7  * purpose with or without fee is hereby granted, provided that the above
8  * copyright notice and this permission notice appear in all copies.
9  *
10 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
11 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
12 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
13 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
14 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
15 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
16 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
17 */
18 #ifdef HAVE_CONFIG_H
19 #include "config.h"
20 #endif
21
22 #include <sys/types.h>
23
24 #include <assert.h>
25 #include <ctype.h>
26 #include <stdint.h>
27 #include <stdio.h>
28 #include <stdlib.h>
29 #include <string.h>
30
31 #include "mandoc.h"
32 #include "out.h"
33 #include "term.h"
34 #include "main.h"
35
36 static void      adjbuf(struct term *p, int);
37 static void      bufferc(struct term *, char);
38 static void      encode(struct term *, const char *, size_t);
39 static void      encodel(struct term *, int);
40
41 void
42 term_free(struct term *p)
43 {
44     if (p->buf)
45         free(p->buf);
46     if (p->syntab)
47         mchars_free(p->syntab);
48     free(p);
49 }
50
51
52 void
53 term_begin(struct term *p, term_margin head,
54            term_margin foot, const void *arg)

```

```

57 {
58     p->headf = head;
59     p->footf = foot;
60     p->argf = arg;
61     (*p->begin)(p);
62 }
63
64 void
65 term_end(struct term *p)
66 {
67     (*p->end)(p);
68 }
69
70 /*
71 * Flush a line of text. A "line" is loosely defined as being something
72 * that should be followed by a newline, regardless of whether it's
73 * broken apart by newlines getting there. A line can also be a
74 * fragment of a columnar list ('Bl -tag' or 'Bl -column'), which does
75 * not have a trailing newline.
76 *
77 * The following flags may be specified:
78 *
79 * - TERMP_NOBREAK: this is the most important and is used when making
80 * columns. In short: don't print a newline and instead expect the
81 * next call to do the padding up to the start of the next column.
82 *
83 * - TERMP_TWOSPACE: make sure there is room for at least two space
84 * characters of padding. Otherwise, rather break the line.
85 *
86 * - TERMP_DANGLE: don't newline when TERMP_NOBREAK is specified and
87 * the line is overrun, and don't pad-right if it's underrun.
88 *
89 * - TERMP_HANG: like TERMP_DANGLE, but doesn't newline when
90 * overrunning, instead save the position and continue at that point
91 * when the next invocation.
92 *
93 * In-line line breaking:
94 *
95 * If TERMP_NOBREAK is specified and the line overruns the right
96 * margin, it will break and pad-right to the right margin after
97 * writing. If maxrmargin is violated, it will break and continue
98 * writing from the right-margin, which will lead to the above scenario
99 * upon exit. Otherwise, the line will break at the right margin.
100 */
101 void
102 term_flushln(struct term *p)
103 {
104     int      i; /* current input position in p->buf */
105     size_t   vis; /* current visual position on output */
106     size_t   vbl; /* number of blanks to prepend to output */
107     size_t   vend; /* end of word visual position on output */
108     size_t   bp; /* visual right border position */
109     size_t   dv; /* temporary for visual pos calculations */
110     int      j; /* temporary loop index for p->buf */
111     int      jhy; /* last hyph before overflow w/r/t j */
112     size_t   maxvis; /* output position of visible boundary */
113     size_t   mmax; /* used in calculating bp */
114
115     /*
116      * First, establish the maximum columns of "visible" content.
117      * This is usually the difference between the right-margin and
118      * an indentation, but can be, for tagged lists or columns, a
119      * small set of values.
120      */

```



```

123  */
124  assert (p->rmargin >= p->offset);
125  dv = p->rmargin - p->offset;
126  maxvis = (int)dv > p->overstep ? dv - (size_t)p->overstep : 0;
127  dv = p->maxrmargin - p->offset;
128  mmax = (int)dv > p->overstep ? dv - (size_t)p->overstep : 0;

130  bp = TERMP_NOBREAK & p->flags ? mmax : maxvis;

132  /*
133  * Calculate the required amount of padding.
134  */
135  vbl = p->offset + p->overstep > p->viscol ?
136  p->offset + p->overstep - p->viscol : 0;

138  vis = vend = 0;
139  i = 0;

141  while (i < p->col) {
142  /*
143  * Handle literal tab characters: collapse all
144  * subsequent tabs into a single huge set of spaces.
145  */
146  while (i < p->col && '\t' == p->buf[i]) {
147  vend = (vis / p->tabwidth + 1) * p->tabwidth;
148  vbl += vend - vis;
149  vis = vend;
150  i++;
151  }

153  /*
154  * Count up visible word characters. Control sequences
155  * (starting with the CSI) aren't counted. A space
156  * generates a non-printing word, which is valid (the
157  * space is printed according to regular spacing rules).
158  */

160  for (j = i, jhy = 0; j < p->col; j++) {
161  if ((j && ' ' == p->buf[j]) || '\t' == p->buf[j])
162  break;

164  /* Back over the the last printed character. */
165  if (8 == p->buf[j]) {
166  assert(j);
167  vend -= (*p->width)(p, p->buf[j - 1]);
168  continue;
169  }

171  /* Regular word. */
172  /* Break at the hyphen point if we overrun. */
173  if (vend > vis && vend < bp &&
174  ASCII_HYPH == p->buf[j])
175  jhy = j;

177  vend += (*p->width)(p, p->buf[j]);
178  }

180  /*
181  * Find out whether we would exceed the right margin.
182  * If so, break to the next line.
183  */
184  if (vend > bp && 0 == jhy && vis > 0) {
185  vend -= vis;
186  (*p->endline)(p);
187  p->viscol = 0;
188  if (TERMP_NOBREAK & p->flags) {

```

```

189  vbl = p->rmargin;
190  vend += p->rmargin - p->offset;
191  } else
192  vbl = p->offset;

194  /* Remove the p->overstep width. */

196  bp += (size_t)p->overstep;
197  p->overstep = 0;
198  }

200  /* Write out the [remaining] word. */
201  for (; i < p->col; i++) {
202  if (vend > bp && jhy > 0 && i > jhy)
203  break;
204  if ('\t' == p->buf[i])
205  break;
206  if (' ' == p->buf[i]) {
207  j = i;
208  while (' ' == p->buf[i])
209  i++;
210  dv = (size_t)(i - j) * (*p->width)(p, ' ');
211  vbl += dv;
212  vend += dv;
213  break;
214  }
215  if (ASCII_NBRSP == p->buf[i]) {
216  vbl += (*p->width)(p, ' ');
217  continue;
218  }

220  /*
221  * Now we definitely know there will be
222  * printable characters to output,
223  * so write preceding white space now.
224  */
225  if (vbl) {
226  (*p->advance)(p, vbl);
227  p->viscol += vbl;
228  vbl = 0;
229  }

231  if (ASCII_HYPH == p->buf[i]) {
232  (*p->letter)(p, '-');
233  p->viscol += (*p->width)(p, '-');
234  continue;
235  }

237  (*p->letter)(p, p->buf[i]);
238  if (8 == p->buf[i])
239  p->viscol -= (*p->width)(p, p->buf[i-1]);
240  else
241  p->viscol += (*p->width)(p, p->buf[i]);
242  }
243  vis = vend;
244  }

246  /*
247  * If there was trailing white space, it was not printed;
248  * so reset the cursor position accordingly.
249  */
250  if (vis)
251  vis -= vbl;

253  p->col = 0;
254  p->overstep = 0;

```

```

256     if ( ! (TERMP_NOBREAK & p->flags) ) {
257         p->viscol = 0;
258         (*p->endline)(p);
259         return;
260     }

262     if (TERMP_HANG & p->flags) {
263         /* We need one blank after the tag. */
264         p->overstep = (int)(vis - maxvis + (*p->width)(p, ' '));

266         /*
267          * Behave exactly the same way as groff:
268          * If we have overstepped the margin, temporarily move
269          * it to the right and flag the rest of the line to be
270          * shorter.
271          * If we landed right at the margin, be happy.
272          * If we are one step before the margin, temporarily
273          * move it one step LEFT and flag the rest of the line
274          * to be longer.
275          */
276         if (p->overstep < -1)
277             p->overstep = 0;
278         return;

280     } else if (TERMP_DANGLE & p->flags)
281         return;

283     /* If the column was overrun, break the line. */
284     if (maxvis <= vis +
285         ((TERMP_TWOSPACE & p->flags) ? (*p->width)(p, ' ') : 0)) {
286         (*p->endline)(p);
287         p->viscol = 0;
288     }
289 }

292 /*
293  * A newline only breaks an existing line; it won't assert vertical
294  * space. All data in the output buffer is flushed prior to the newline
295  * assertion.
296  */
297 void
298 term_newln(struct term *p)
299 {

301     p->flags |= TERMP_NOSPACE;
302     if (p->col || p->viscol)
303         term_flushln(p);
304 }

307 /*
308  * Asserts a vertical space (a full, empty line-break between lines).
309  * Note that if used twice, this will cause two blank spaces and so on.
310  * All data in the output buffer is flushed prior to the newline
311  * assertion.
312  */
313 void
314 term_vspace(struct term *p)
315 {

317     term_newln(p);
318     p->viscol = 0;
319     (*p->endline)(p);
320 }

```

```

322 void
323 term_fontlast(struct term *p)
324 {
325     enum termfont    f;

327     f = p->fontl;
328     p->fontl = p->fontq[p->fonti];
329     p->fontq[p->fonti] = f;
330 }

333 void
334 term_fontrepl(struct term *p, enum termfont f)
335 {

337     p->fontl = p->fontq[p->fonti];
338     p->fontq[p->fonti] = f;
339 }

342 void
343 term_fontpush(struct term *p, enum termfont f)
344 {

346     assert(p->fonti + 1 < 10);
347     p->fontl = p->fontq[p->fonti];
348     p->fontq[++p->fonti] = f;
349 }

352 const void *
353 term_fontq(struct term *p)
354 {

356     return(&p->fontq[p->fonti]);
357 }

360 enum termfont
361 term_fonttop(struct term *p)
362 {

364     return(p->fontq[p->fonti]);
365 }

368 void
369 term_fontpopq(struct term *p, const void *key)
370 {

372     while (p->fonti >= 0 && key != &p->fontq[p->fonti])
373         p->fonti--;
374     assert(p->fonti >= 0);
375 }

378 void
379 term_fontpop(struct term *p)
380 {

382     assert(p->fonti);
383     p->fonti--;
384 }

386 /*

```

```

387 * Handle pwords, partial words, which may be either a single word or a
388 * phrase that cannot be broken down (such as a literal string). This
389 * handles word styling.
390 */
391 void
392 term_word(struct term *p, const char *word)
393 {
394     const char    *seq, *cp;
395     char          c;
396     int           sz, uc;
397     size_t       ssz;
398     enum mandoc_esc  esc;
399
400     if ( ! (TERMP_NOSPACE & p->flags) ) {
401         if ( ! (TERMP_KEEP & p->flags) ) {
402             if (TERMP_PREKEEP & p->flags)
403                 p->flags |= TERMP_KEEP;
404             bufferc(p, ' ');
405             if (TERMP_SENTENCE & p->flags)
406                 bufferc(p, ' ');
407         } else
408             bufferc(p, ASCII_NBRSP);
409     }
410
411     if ( ! (p->flags & TERMP_NONOSPACE) )
412         p->flags &= ~TERMP_NOSPACE;
413     else
414         p->flags |= TERMP_NOSPACE;
415
416     p->flags &= ~(TERMP_SENTENCE | TERMP_IGNDELIM);
417
418     while ('\\0' != *word) {
419         if ((ssz = strcspn(word, "\\\")) > 0)
420             encode(p, word, ssz);
421
422         word += (int)ssz;
423         if ('\\' != *word)
424             continue;
425
426         word++;
427         esc = mandoc_escape(&word, &seq, &sz);
428         if (ESCAPE_ERROR == esc)
429             break;
430
431         if (TERMENC_ASCII != p->enc)
432             switch (esc) {
433                 case (ESCAPE_UNICODE):
434                     uc = mchars_num2uc(seq + 1, sz - 1);
435                     if ('\\0' == uc)
436                         break;
437                     encodel(p, uc);
438                     continue;
439                 case (ESCAPE_SPECIAL):
440                     uc = mchars_spec2cp(p->syntab, seq, sz);
441                     if (uc <= 0)
442                         break;
443                     encodel(p, uc);
444                     continue;
445                 default:
446                     break;
447             }
448
449         switch (esc) {
450             case (ESCAPE_UNICODE):
451                 encodel(p, '?');
452                 break;

```

```

453         case (ESCAPE_NUMBERED):
454             c = mchars_num2char(seq, sz);
455             if ('\\0' != c)
456                 encode(p, &c, 1);
457             break;
458         case (ESCAPE_SPECIAL):
459             cp = mchars_spec2str(p->syntab, seq, sz, &ssz);
460             if (NULL != cp)
461                 encode(p, cp, ssz);
462             else if (1 == ssz)
463                 encode(p, seq, sz);
464             break;
465         case (ESCAPE_FONTBOLD):
466             term_fontrepl(p, TERMFONT_BOLD);
467             break;
468         case (ESCAPE_FONTITALIC):
469             term_fontrepl(p, TERMFONT_UNDER);
470             break;
471         case (ESCAPE_FONT):
472             /* FALLTHROUGH */
473         case (ESCAPE_FONTROMAN):
474             term_fontrepl(p, TERMFONT_NONE);
475             break;
476         case (ESCAPE_FONTPREV):
477             term_fontlast(p);
478             break;
479         case (ESCAPE_NOSPACE):
480             if ('\\0' == *word)
481                 p->flags |= TERMP_NOSPACE;
482             break;
483         default:
484             break;
485     }
486 }
487 }
488
489 static void
490 adjbuf(struct term *p, int sz)
491 {
492     if (0 == p->maxcols)
493         p->maxcols = 1024;
494     while (sz >= p->maxcols)
495         p->maxcols <<= 2;
496
497     p->buf = mandoc_realloc
498         (p->buf, sizeof(int) * (size_t)p->maxcols);
499 }
500
501 static void
502 bufferc(struct term *p, char c)
503 {
504     if (p->col + 1 >= p->maxcols)
505         adjbuf(p, p->col + 1);
506
507     p->buf[p->col++] = c;
508 }
509
510 /*
511  * See encode().
512  * Do this for a single (probably unicode) value.
513  * Does not check for non-decorated glyphs.
514  */
515 static void
516 encodel(struct term *p, int c)

```



```

651         if ('\0' != c)
652             sz += (*p->width)(p, c);
653         break;
654     case (ESCAPE_SPECIAL):
655         rhs = mchars_spec2str
656             (p->symtab, seq, ssz, &rsz);
657
658         if (ssz != 1 || rhs)
659             break;
660
661         rhs = seq;
662         rsz = ssz;
663         break;
664     default:
665         break;
666 }
667
668 if (NULL == rhs)
669     break;
670
671 for (i = 0; i < rsz; i++)
672     sz += (*p->width)(p, *rhs++);
673 break;
674 case (ASCII_NBRSP):
675     sz += (*p->width)(p, ' ');
676     cp++;
677     break;
678 case (ASCII_HYPH):
679     sz += (*p->width)(p, '-');
680     cp++;
681     break;
682 default:
683     break;
684 }
685 }
686
687 return(sz);
688 }
689
690 /* ARGSUSED */
691 size_t
692 term_vspan(const struct term *p, const struct roffsu *su)
693 {
694     double    r;
695
696     switch (su->unit) {
697     case (SCALE_CM):
698         r = su->scale * 2;
699         break;
700     case (SCALE_IN):
701         r = su->scale * 6;
702         break;
703     case (SCALE_PC):
704         r = su->scale;
705         break;
706     case (SCALE_PT):
707         r = su->scale / 8;
708         break;
709     case (SCALE_MM):
710         r = su->scale / 1000;
711         break;
712     case (SCALE_VS):
713         r = su->scale;
714         break;
715     default:
716         r = su->scale - 1;

```

```

717         break;
718     }
719
720     if (r < 0.0)
721         r = 0.0;
722     return(/* LINTED */(size_t)
723         r);
724 }
725
726 size_t
727 term_hspan(const struct term *p, const struct roffsu *su)
728 {
729     double    v;
730
731     v = ((*p->hspan)(p, su));
732     if (v < 0.0)
733         v = 0.0;
734     return((size_t) /* LINTED */
735         v);
736 }

```

```

*****
4549 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/term.h
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: term.h,v 1.90 2011/12/04 23:10:52 schwarze Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifndef TERM_H
18 #define TERM_H
19
20 __BEGIN_DECLS
21
22 struct term;
23
24 enum termenc {
25     TERMENC_ASCII,
26     TERMENC_LOCALE,
27     TERMENC_UTF8
28 };
29
30 enum termtype {
31     TERMTYPE_CHAR,
32     TERMTYPE_PS,
33     TERMTYPE_PDF
34 };
35
36 enum termfont {
37     TERMFONT_NONE = 0,
38     TERMFONT_BOLD,
39     TERMFONT_UNDER,
40     TERMFONT_MAX
41 };
42
43 #define TERM_MAXMARGIN 100000 /* FIXME */
44
45 typedef void (*term_margin)(struct term *, const void *);
46
47 struct term_tbl {
48     int width; /* width in fixed chars */
49     int decimal; /* decimal point position */
50 };
51
52 struct term {
53     enum termtype type;
54     struct termtbl tbl; /* table configuration */
55     int mdocstyle; /* imitate mdoc(7) output */
56     size_t defindent; /* Default indent for text. */

```

```

57     size_t defrmargin; /* Right margin of the device. */
58     size_t rmargin; /* Current right margin. */
59     size_t maxrmargin; /* Max right margin. */
60     int maxcols; /* Max size of buf. */
61     size_t offset; /* Margin offset. */
62     size_t tabwidth; /* Distance of tab positions. */
63     int col; /* Bytes in buf. */
64     size_t viscol; /* Chars on current line. */
65     int overstep; /* See term_flushln(). */
66     int flags;
67 #define TERMP_SENTENCE (1 << 1) /* Space before a sentence. */
68 #define TERMP_NOSPACE (1 << 2) /* No space before words. */
69 #define TERMP_NOBREAK (1 << 4) /* See term_flushln(). */
70 #define TERMP_IGNDELIM (1 << 6) /* Delims like regulars. */
71 #define TERMP_NONOSPACE (1 << 7) /* No space (no autounset). */
72 #define TERMP_DANGLE (1 << 8) /* See term_flushln(). */
73 #define TERMP_HANG (1 << 9) /* See term_flushln(). */
74 #define TERMP_TWOSPACE (1 << 10) /* See term_flushln(). */
75 #define TERMP_NOSPLIT (1 << 11) /* See term_an_pre/post(). */
76 #define TERMP_SPLIT (1 << 12) /* See term_an_pre/post(). */
77 #define TERMP_ANPREC (1 << 13) /* See term_an_pre(). */
78 #define TERMP_KEEP (1 << 14) /* Keep words together. */
79 #define TERMP_PREKEEP (1 << 15) /* ...starting with the next one. */
80     int *buf; /* Output buffer. */
81     enum termenc enc; /* Type of encoding. */
82     struct mchars *symtab; /* Encoded-symbol table. */
83     enum termfont fontl; /* Last font set. */
84     enum termfont fontq[10]; /* Symmetric fonts. */
85     int fonti; /* Index of font stack. */
86     term_margin headf; /* invoked to print head */
87     term_margin footf; /* invoked to print foot */
88     void (*letter)(struct term *, int);
89     void (*begin)(struct term *);
90     void (*end)(struct term *);
91     void (*endline)(struct term *);
92     void (*advance)(struct term *, size_t);
93     size_t (*width)(const struct term *, int);
94     double (*hspan)(const struct term *,
95         const struct roffsu *);
96     const void *argf; /* arg for headf/footf */
97     struct term_ps *ps;
98 };
99
100 void term_eqn(struct term *, const struct eqn *);
101 void term_tbl(struct term *, const struct tbl_span *);
102 void term_free(struct term *);
103 void term_newln(struct term *);
104 void term_vspace(struct term *);
105 void term_word(struct term *, const char *);
106 void term_flushln(struct term *);
107 void term_begin(struct term *, term_margin,
108     term_margin, const void *);
109 void term_end(struct term *);
110
111 size_t term_hspan(const struct term *,
112     const struct roffsu *);
113 size_t term_vspan(const struct term *,
114     const struct roffsu *);
115 size_t term_strlen(const struct term *, const char *);
116 size_t term_len(const struct term *, size_t);
117
118 enum termfont term_fonttop(struct term *);
119 const void *term_fontq(struct term *);
120 void term_fontpush(struct term *, enum termfont);
121 void term_fontpop(struct term *);
122 void term_fontpopq(struct term *, const void *);

```

```
123 void          term_fontrepl(struct term *, enum termfont);
124 void          term_fontlast(struct term *);

126 __END_DECLS

128 #endif /*!TERM_H*/
```

```

*****
5403 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/term_ascii.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: term_ascii.c,v 1.20 2011/12/04 23:10:52 schwarze Exp $ */
2 /*
3  * Copyright (c) 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <sys/types.h>
22
23 #include <assert.h>
24 #ifdef USE_WCHAR
25 #include <locale.h>
26 #endif
27 #include <stdint.h>
28 #include <stdio.h>
29 #include <stdlib.h>
30 #include <unistd.h>
31 #ifdef USE_WCHAR
32 #include <wchar.h>
33 #endif
34
35 #include "mandoc.h"
36 #include "out.h"
37 #include "term.h"
38 #include "main.h"
39
40 /*
41  * Sadly, this doesn't seem to be defined on systems even when they
42  * support it. For the time being, remove it and let those compiling
43  * the software decide for themselves what to use.
44  */
45 #if 0
46 #if ! defined(__STDC_ISO_10646__)
47 #undef USE_WCHAR
48 #endif
49 #endif
50
51 static struct termp *ascii_init(enum termenc, char *);
52 static double ascii_hspan(const struct termp *,
53     const struct roffsu *);
54 static size_t ascii_width(const struct termp *, int);
55 static void ascii_advance(struct termp *, size_t);
56 static void ascii_begin(struct termp *);

```

```

57 static void ascii_end(struct termp *);
58 static void ascii_endline(struct termp *);
59 static void ascii_letter(struct termp *, int);
60
61 #ifdef USE_WCHAR
62 static void locale_advance(struct termp *, size_t);
63 static void locale_endline(struct termp *);
64 static void locale_letter(struct termp *, int);
65 static size_t locale_width(const struct termp *, int);
66 #endif
67
68 static struct termp *
69 ascii_init(enum termenc enc, char *outopts)
70 {
71     const char *toks[4];
72     char *v;
73     struct termp *p;
74
75     p = mandoc_malloc(1, sizeof(struct termp));
76     p->enc = enc;
77
78     p->tabwidth = 5;
79     p->defrmargin = 78;
80
81     p->begin = ascii_begin;
82     p->end = ascii_end;
83     p->hspan = ascii_hspan;
84     p->type = TERMTYPE_CHAR;
85
86     p->enc = TERMENC_ASCII;
87     p->advance = ascii_advance;
88     p->endline = ascii_endline;
89     p->letter = ascii_letter;
90     p->width = ascii_width;
91
92 #ifdef USE_WCHAR
93     if (TERMENC_ASCII != enc) {
94         v = TERMENC_LOCALE == enc ?
95             setlocale(LC_ALL, "") :
96             setlocale(LC_CTYPE, "UTF-8");
97         if (NULL != v && MB_CUR_MAX > 1) {
98             p->enc = enc;
99             p->advance = locale_advance;
100            p->endline = locale_endline;
101            p->letter = locale_letter;
102            p->width = locale_width;
103        }
104    }
105 #endif
106
107     toks[0] = "indent";
108     toks[1] = "width";
109     toks[2] = "mdoc";
110     toks[3] = NULL;
111
112     while (outopts && *outopts)
113         switch (getsubopt(&outopts, UNCONST(toks), &v)) {
114             case (0):
115                 p->defindent = (size_t)atoi(v);
116                 break;
117             case (1):
118                 p->defrmargin = (size_t)atoi(v);
119                 break;
120             case (2):
121                 /*
122                  * Temporary, undocumented mode

```



```

123         * to imitate mdoc(7) output style.
124         */
125         p->mdocstyle = 1;
126         p->defindent = 5;
127         break;
128     default:
129         break;
130     }

132     /* Enforce a lower boundary. */
133     if (p->defmargin < 58)
134         p->defmargin = 58;

136     return(p);
137 }

139 void *
140 ascii_alloc(char *outopts)
141 {

143     return(ascii_init(TERMENC_ASCII, outopts));
144 }

146 void *
147 utf8_alloc(char *outopts)
148 {

150     return(ascii_init(TERMENC_UTF8, outopts));
151 }

154 void *
155 locale_alloc(char *outopts)
156 {

158     return(ascii_init(TERMENC_LOCALE, outopts));
159 }

161 /* ARGSUSED */
162 static size_t
163 ascii_width(const struct term *p, int c)
164 {

166     return(1);
167 }

169 void
170 ascii_free(void *arg)
171 {

173     term_free((struct term *)arg);
174 }

176 /* ARGSUSED */
177 static void
178 ascii_letter(struct term *p, int c)
179 {
180     putchar(c);
181 }
182 }

184 static void
185 ascii_begin(struct term *p)
186 {

188     (*p->headf)(p, p->argf);

```

```

189 }

191 static void
192 ascii_end(struct term *p)
193 {

195     (*p->footf)(p, p->argf);
196 }

198 /* ARGSUSED */
199 static void
200 ascii_endline(struct term *p)
201 {

203     putchar('\n');
204 }

206 /* ARGSUSED */
207 static void
208 ascii_advance(struct term *p, size_t len)
209 {
210     size_t     i;

212     for (i = 0; i < len; i++)
213         putchar(' ');
214 }

216 /* ARGSUSED */
217 static double
218 ascii_hspan(const struct term *p, const struct roffsu *su)
219 {
220     double     r;

222     /*
223      * Approximate based on character width. These are generated
224      * entirely by eyeballing the screen, but appear to be correct.
225      */

227     switch (su->unit) {
228     case (SCALE_CM):
229         r = 4 * su->scale;
230         break;
231     case (SCALE_IN):
232         r = 10 * su->scale;
233         break;
234     case (SCALE_PC):
235         r = (10 * su->scale) / 6;
236         break;
237     case (SCALE_PT):
238         r = (10 * su->scale) / 72;
239         break;
240     case (SCALE_MM):
241         r = su->scale / 1000;
242         break;
243     case (SCALE_VS):
244         r = su->scale * 2 - 1;
245         break;
246     default:
247         r = su->scale;
248         break;
249     }

251     return(r);
252 }

254 #ifndef USE_WCHAR

```

```
255 /* ARGSUSED */
256 static size_t
257 locale_width(const struct term *p, int c)
258 {
259     int rc;
260
261     return((rc = wcwidth(c)) < 0 ? 0 : rc);
262 }
263
264 /* ARGSUSED */
265 static void
266 locale_advance(struct term *p, size_t len)
267 {
268     size_t i;
269
270     for (i = 0; i < len; i++)
271         putwchar(L' ');
272 }
273
274 /* ARGSUSED */
275 static void
276 locale_endline(struct term *p)
277 {
278     putwchar(L'\n');
279 }
280
281
282 /* ARGSUSED */
283 static void
284 locale_letter(struct term *p, int c)
285 {
286     putwchar(c);
287 }
288
289 #endif
```

```

*****
23551 Sat Jul 19 19:24:32 2014
new/usr/src/cmd/mandoc/term_ps.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 /* $Id: term_ps.c,v 1.54 2011/10/16 12:20:34 schwarze Exp $ */
2 /*
3  * Copyright (c) 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <sys/types.h>
22
23 #include <assert.h>
24 #include <stdarg.h>
25 #include <stdint.h>
26 #include <stdio.h>
27 #include <stdlib.h>
28 #include <string.h>
29 #include <time.h>
30 #include <unistd.h>
31
32 #include "mandoc.h"
33 #include "out.h"
34 #include "main.h"
35 #include "term.h"
36
37 /* These work the buffer used by the header and footer. */
38 #define PS_BUFSLOP      128
39
40 /* Convert PostScript point "x" to an AFM unit. */
41 #define PNT2AFM(p, x) /* LINTED */ \
42     (size_t)((double)(x) * (1000.0 / (double)(p)->ps->scale))
43
44 /* Convert an AFM unit "x" to a PostScript points */
45 #define AFM2PNT(p, x) /* LINTED */ \
46     ((double)(x) / (1000.0 / (double)(p)->ps->scale))
47
48 struct glyph {
49     unsigned short    wx; /* WX in AFM */
50 };
51
52 struct font {
53     const char        *name; /* FontName in AFM */
54     #define MAXCHAR    95 /* total characters we can handle */
55     struct glyph      gly[MAXCHAR]; /* glyph metrics */
56 };

```

```

58 struct term_ps {
59     int                flags;
60     #define PS_INLINE    (1 << 0) /* we're in a word */
61     #define PS_MARGINS  (1 << 1) /* we're in the margins */
62     #define PS_NEWPAGE  (1 << 2) /* new page, no words yet */
63     size_t            pscol; /* visible column (AFM units) */
64     size_t            psrow; /* visible row (AFM units) */
65     char              *psmarg; /* margin buf */
66     size_t            psmargsz; /* margin buf size */
67     size_t            psmargcur; /* cur index in margin buf */
68     char              last; /* character buffer */
69     enum termfont     lastf; /* last set font */
70     size_t            scale; /* font scaling factor */
71     size_t            pages; /* number of pages shown */
72     size_t            lineheight; /* line height (AFM units) */
73     size_t            top; /* body top (AFM units) */
74     size_t            bottom; /* body bottom (AFM units) */
75     size_t            height; /* page height (AFM units) */
76     size_t            width; /* page width (AFM units) */
77     size_t            left; /* body left (AFM units) */
78     size_t            header; /* header pos (AFM units) */
79     size_t            footer; /* footer pos (AFM units) */
80     size_t            pdfbytes; /* current output byte */
81     size_t            pdflastpg; /* byte of last page mark */
82     size_t            pdfbody; /* start of body object */
83     size_t            *pdfobjjs; /* table of object offsets */
84     size_t            pdfobjjsz; /* size of pdfobjjs */
85 };
86
87 static double        ps_hspan(const struct term *t,
88                               const struct roffsu *r);
89 static size_t        ps_width(const struct term *t, int);
90 static void          ps_advance(struct term *t, size_t);
91 static void          ps_begin(struct term *t);
92 static void          ps_closepage(struct term *t);
93 static void          ps_end(struct term *t);
94 static void          ps_endline(struct term *t);
95 static void          ps_fclose(struct term *t);
96 static void          ps_growbuf(struct term *t, size_t);
97 static void          ps_letter(struct term *t, int);
98 static void          ps_pclose(struct term *t);
99 static void          ps_pletter(struct term *t, int);
100 static void          ps_printf(struct term *t, const char *, ...);
101 static void          ps_putchar(struct term *t, char);
102 static void          ps_setfont(struct term *t, enum termfont);
103 static struct term   *pspdf_alloc(char *);
104 static void          pdf_obj(struct term *t, size_t);
105
106 /*
107  * We define, for the time being, three fonts: bold, oblique/italic, and
108  * normal (roman). The following table hard-codes the font metrics for
109  * ASCII, i.e., 32--127.
110  */
111
112 static const struct font fonts[TERMFONT_MAX] = {
113     { "Times-Roman", {
114         { 250 },
115         { 333 },
116         { 408 },
117         { 500 },
118         { 500 },
119         { 833 },
120         { 778 },
121         { 333 },
122         { 333 },

```

```

123          { 333 ,
124          { 500 ,
125          { 564 ,
126          { 250 ,
127          { 333 ,
128          { 250 ,
129          { 278 ,
130          { 500 ,
131          { 500 ,
132          { 500 ,
133          { 500 ,
134          { 500 ,
135          { 500 ,
136          { 500 ,
137          { 500 ,
138          { 500 ,
139          { 500 ,
140          { 278 ,
141          { 278 ,
142          { 564 ,
143          { 564 ,
144          { 564 ,
145          { 444 ,
146          { 921 ,
147          { 722 ,
148          { 667 ,
149          { 667 ,
150          { 722 ,
151          { 611 ,
152          { 556 ,
153          { 722 ,
154          { 722 ,
155          { 333 ,
156          { 389 ,
157          { 722 ,
158          { 611 ,
159          { 889 ,
160          { 722 ,
161          { 722 ,
162          { 556 ,
163          { 722 ,
164          { 667 ,
165          { 556 ,
166          { 611 ,
167          { 722 ,
168          { 722 ,
169          { 944 ,
170          { 722 ,
171          { 722 ,
172          { 611 ,
173          { 333 ,
174          { 278 ,
175          { 333 ,
176          { 469 ,
177          { 500 ,
178          { 333 ,
179          { 444 ,
180          { 500 ,
181          { 444 ,
182          { 500 ,
183          { 444 ,
184          { 333 ,
185          { 500 ,
186          { 500 ,
187          { 278 ,
188          { 278 ,

```

```

189          { 500 ,
190          { 278 ,
191          { 778 ,
192          { 500 ,
193          { 500 ,
194          { 500 ,
195          { 500 ,
196          { 333 ,
197          { 389 ,
198          { 278 ,
199          { 500 ,
200          { 500 ,
201          { 722 ,
202          { 500 ,
203          { 500 ,
204          { 444 ,
205          { 480 ,
206          { 200 ,
207          { 480 ,
208          { 541 ,
209          } ,
210          { "Times-Bold", {
211          { 250 ,
212          { 333 ,
213          { 555 ,
214          { 500 ,
215          { 500 ,
216          { 1000 ,
217          { 833 ,
218          { 333 ,
219          { 333 ,
220          { 333 ,
221          { 500 ,
222          { 570 ,
223          { 250 ,
224          { 333 ,
225          { 250 ,
226          { 278 ,
227          { 500 ,
228          { 500 ,
229          { 500 ,
230          { 500 ,
231          { 500 ,
232          { 500 ,
233          { 500 ,
234          { 500 ,
235          { 500 ,
236          { 500 ,
237          { 333 ,
238          { 333 ,
239          { 570 ,
240          { 570 ,
241          { 570 ,
242          { 500 ,
243          { 930 ,
244          { 722 ,
245          { 667 ,
246          { 722 ,
247          { 722 ,
248          { 667 ,
249          { 611 ,
250          { 778 ,
251          { 778 ,
252          { 389 ,
253          { 500 ,
254          { 778 ,

```

```

255      { 667 },
256      { 944 },
257      { 722 },
258      { 778 },
259      { 611 },
260      { 778 },
261      { 722 },
262      { 556 },
263      { 667 },
264      { 722 },
265      { 722 },
266      { 1000 },
267      { 722 },
268      { 722 },
269      { 667 },
270      { 333 },
271      { 278 },
272      { 333 },
273      { 581 },
274      { 500 },
275      { 333 },
276      { 500 },
277      { 556 },
278      { 444 },
279      { 556 },
280      { 444 },
281      { 333 },
282      { 500 },
283      { 556 },
284      { 278 },
285      { 333 },
286      { 556 },
287      { 278 },
288      { 833 },
289      { 556 },
290      { 500 },
291      { 556 },
292      { 556 },
293      { 444 },
294      { 389 },
295      { 333 },
296      { 556 },
297      { 500 },
298      { 722 },
299      { 500 },
300      { 500 },
301      { 444 },
302      { 394 },
303      { 220 },
304      { 394 },
305      { 520 },
306      },
307      { "Times-Italic", {
308          { 250 },
309          { 333 },
310          { 420 },
311          { 500 },
312          { 500 },
313          { 833 },
314          { 778 },
315          { 333 },
316          { 333 },
317          { 333 },
318          { 500 },
319          { 675 },
320          { 250 },

```

```

321      { 333 },
322      { 250 },
323      { 278 },
324      { 500 },
325      { 500 },
326      { 500 },
327      { 500 },
328      { 500 },
329      { 500 },
330      { 500 },
331      { 500 },
332      { 500 },
333      { 500 },
334      { 333 },
335      { 333 },
336      { 675 },
337      { 675 },
338      { 675 },
339      { 500 },
340      { 920 },
341      { 611 },
342      { 611 },
343      { 667 },
344      { 722 },
345      { 611 },
346      { 611 },
347      { 722 },
348      { 722 },
349      { 333 },
350      { 444 },
351      { 667 },
352      { 556 },
353      { 833 },
354      { 667 },
355      { 722 },
356      { 611 },
357      { 722 },
358      { 611 },
359      { 500 },
360      { 556 },
361      { 722 },
362      { 611 },
363      { 833 },
364      { 611 },
365      { 556 },
366      { 556 },
367      { 389 },
368      { 278 },
369      { 389 },
370      { 422 },
371      { 500 },
372      { 333 },
373      { 500 },
374      { 500 },
375      { 444 },
376      { 500 },
377      { 444 },
378      { 278 },
379      { 500 },
380      { 500 },
381      { 278 },
382      { 278 },
383      { 444 },
384      { 278 },
385      { 722 },
386      { 500 },

```

```

387     { 500 },
388     { 500 },
389     { 500 },
390     { 389 },
391     { 389 },
392     { 278 },
393     { 500 },
394     { 444 },
395     { 667 },
396     { 444 },
397     { 444 },
398     { 389 },
399     { 400 },
400     { 275 },
401     { 400 },
402     { 541 },
403     } },
404 };

406 void *
407 pdf_alloc(char *outopts)
408 {
409     struct term *p;

411     if (NULL != (p = pspdf_alloc(outopts)))
412         p->type = TERMTYPE_PDF;

414     return(p);
415 }

417 void *
418 ps_alloc(char *outopts)
419 {
420     struct term *p;

422     if (NULL != (p = pspdf_alloc(outopts)))
423         p->type = TERMTYPE_PS;

425     return(p);
426 }

428 static struct term *
429 pspdf_alloc(char *outopts)
430 {
431     struct term *p;
432     unsigned int pagex, pagey;
433     size_t marginx, marginy, lineheight;
434     const char *toks[2];
435     const char *pp;
436     char *v;

438     p = mandoc_calloc(1, sizeof(struct term));
439     p->enc = TERMENC_ASCII;
440     p->ps = mandoc_calloc(1, sizeof(struct term_ps));

442     p->advance = ps_advance;
443     p->begin = ps_begin;
444     p->end = ps_end;
445     p->endline = ps_endline;
446     p->hspan = ps_hspan;
447     p->letter = ps_letter;
448     p->width = ps_width;
449
450     toks[0] = "paper";
451     toks[1] = NULL;

```

```

453     pp = NULL;

455     while (outopts && *outopts)
456         switch (getsubopt(&outopts, UNCONST(toks), &v)) {
457             case (0):
458                 pp = v;
459                 break;
460             default:
461                 break;
462         }

464     /* Default to US letter (millimetres). */

466     pagex = 216;
467     pagey = 279;

469     /*
470     * The ISO-269 paper sizes can be calculated automatically, but
471     * it would require bringing in -lm for pow() and I'd rather not
472     * do that. So just do it the easy way for now. Since this
473     * only happens once, I'm not terribly concerned.
474     */

476     if (pp && strcasecmp(pp, "letter")) {
477         if (0 == strcasecmp(pp, "a3")) {
478             pagex = 297;
479             pagey = 420;
480         } else if (0 == strcasecmp(pp, "a4")) {
481             pagex = 210;
482             pagey = 297;
483         } else if (0 == strcasecmp(pp, "a5")) {
484             pagex = 148;
485             pagey = 210;
486         } else if (0 == strcasecmp(pp, "legal")) {
487             pagex = 216;
488             pagey = 356;
489         } else if (2 != sscanf(pp, "%ux%u", &pagex, &pagey))
490             fprintf(stderr, "%s: Unknown paper\n", pp);
491     }

493     /*
494     * This MUST be defined before any PNT2AFM or AFM2PNT
495     * calculations occur.
496     */

498     p->ps->scale = 11;

500     /* Remember millimetres -> AFM units. */

502     pagex = PNT2AFM(p, ((double)pagex * 2.834));
503     pagey = PNT2AFM(p, ((double)pagey * 2.834));

505     /* Margins are 1/9 the page x and y. */

507     marginx = /* LINTED */
508              (size_t)((double)pagex / 9.0);
509     marginy = /* LINTED */
510              (size_t)((double)pagey / 9.0);

512     /* Line-height is 1.4em. */

514     lineheight = PNT2AFM(p, ((double)p->ps->scale * 1.4));

516     p->ps->width = (size_t)pagex;
517     p->ps->height = (size_t)pagey;
518     p->ps->header = pagey - (marginx / 2) - (lineheight / 2);

```

```

519     p->ps->top = pagey - marginy;
520     p->ps->footer = (marginy / 2) - (lineheight / 2);
521     p->ps->bottom = marginy;
522     p->ps->left = marginx;
523     p->ps->lineheight = lineheight;

525     p->defmargin = pagex - (marginx * 2);
526     return(p);
527 }

530 void
531 pspdf_free(void *arg)
532 {
533     struct term *p;

535     p = (struct term *)arg;

537     if (p->ps->psmarg)
538         free(p->ps->psmarg);
539     if (p->ps->pdfobjs)
540         free(p->ps->pdfobjs);

542     free(p->ps);
543     term_free(p);
544 }

547 static void
548 ps_printf(struct term *p, const char *fmt, ...)
549 {
550     va_list      ap;
551     int          pos, len;

553     va_start(ap, fmt);

555     /*
556      * If we're running in regular mode, then pipe directly into
557      * vprintf().  If we're processing margins, then push the data
558      * into our growable margin buffer.
559      */

561     if ( ! (PS_MARGINS & p->ps->flags)) {
562         len = vprintf(fmt, ap);
563         va_end(ap);
564         p->ps->pdfbytes += /* LINTED */
565             len < 0 ? 0 : (size_t)len;
566         return;
567     }

569     /*
570      * XXX: I assume that the in-margin print won't exceed
571      * PS_BUFSLOP (128 bytes), which is reasonable but still an
572      * assumption that will cause pukeage if it's not the case.
573      */

575     ps_growbuf(p, PS_BUFSLOP);

577     pos = (int)p->ps->psmargcur;
578     vsnprintf(&p->ps->psmarg[pos], PS_BUFSLOP, fmt, ap);

580     va_end(ap);

582     p->ps->psmargcur = strlen(p->ps->psmarg);
583 }

```

```

586 static void
587 ps_putchar(struct term *p, char c)
588 {
589     int          pos;

591     /* See ps_printf(). */

593     if ( ! (PS_MARGINS & p->ps->flags)) {
594         /* LINTED */
595         putchar(c);
596         p->ps->pdfbytes++;
597         return;
598     }

600     ps_growbuf(p, 2);

602     pos = (int)p->ps->psmargcur++;
603     p->ps->psmarg[pos++] = c;
604     p->ps->psmarg[pos] = '\0';
605 }

608 static void
609 pdf_obj(struct term *p, size_t obj)
610 {
612     assert(obj > 0);

614     if ((obj - 1) >= p->ps->pdfobjsz) {
615         p->ps->pdfobjsz = obj + 128;
616         p->ps->pdfobjs = realloc
617             (p->ps->pdfobjs,
618              p->ps->pdfobjsz * sizeof(size_t));
619         if (NULL == p->ps->pdfobjs) {
620             perror(NULL);
621             exit((int)MANDOCLEVEL_SYSERR);
622         }
623     }

625     p->ps->pdfobjs[(int)obj - 1] = p->ps->pdfbytes;
626     ps_printf(p, "%zu 0 obj\n", obj);
627 }

630 static void
631 ps_closepage(struct term *p)
632 {
633     int          i;
634     size_t       len, base;

636     /*
637      * Close out a page that we've already flushed to output.  In
638      * PostScript, we simply note that the page must be shown.  In
639      * PDF, we must now create the Length, Resource, and Page node
640      * for the page contents.
641      */

643     assert(p->ps->psmarg && p->ps->psmarg[0]);
644     ps_printf(p, "%s", p->ps->psmarg);

646     if (TERMTYPE_PS != p->type) {
647         ps_printf(p, "ET\n");

649         len = p->ps->pdfbytes - p->ps->pdflastpg;
650         base = p->ps->pages * 4 + p->ps->pdfbody;

```

```

652         ps_printf(p, "endstream\nendobj\n");
654
655         /* Length of content. */
656         pdf_obj(p, base + 1);
657         ps_printf(p, "%zu\nendobj\n", len);
658
659         /* Resource for content. */
660         pdf_obj(p, base + 2);
661         ps_printf(p, "<<\n/ProcSet [/PDF /Text]\n");
662         ps_printf(p, "/Font <<\n");
663         for (i = 0; i < (int)TERMFONT_MAX; i++)
664             ps_printf(p, "/F%d %d 0 R\n", i, 3 + i);
665         ps_printf(p, ">>\n>>\n");
666
667         /* Page node. */
668         pdf_obj(p, base + 3);
669         ps_printf(p, "<<\n");
670         ps_printf(p, "/Type /Page\n");
671         ps_printf(p, "/Parent 2 0 R\n");
672         ps_printf(p, "/Resources %zu 0 R\n", base + 2);
673         ps_printf(p, "/Contents %zu 0 R\n", base);
674         ps_printf(p, ">>\nendobj\n");
675     } else
676         ps_printf(p, "showpage\n");
677
678     p->ps->pages++;
679     p->ps->psrow = p->ps->top;
680     assert(! (PS_NEWPAGE & p->ps->flags));
681     p->ps->flags |= PS_NEWPAGE;
682 }
683
684 /* ARGSUSED */
685 static void
686 ps_end(struct term *p)
687 {
688     size_t      i, xref, base;
689
690     /*
691     * At the end of the file, do one last showpage. This is the
692     * same behaviour as groff(1) and works for multiple pages as
693     * well as just one.
694     */
695
696     if (! (PS_NEWPAGE & p->ps->flags)) {
697         assert(0 == p->ps->flags);
698         assert('\0' == p->ps->last);
699         ps_closepage(p);
700     }
701
702     if (TERMTYPE_PS == p->type) {
703         ps_printf(p, "%%Trailer\n");
704         ps_printf(p, "%%Pages: %zu\n", p->ps->pages);
705         ps_printf(p, "%%EOF\n");
706         return;
707     }
708
709     pdf_obj(p, 2);
710     ps_printf(p, "<<\n/Type /Pages\n");
711     ps_printf(p, "/MediaBox [0 0 %zu %zu]\n",
712              (size_t)AFM2PNT(p, p->ps->width),
713              (size_t)AFM2PNT(p, p->ps->height));
714
715     ps_printf(p, "/Count %zu\n", p->ps->pages);
716     ps_printf(p, "/Kids [");

```

```

718         for (i = 0; i < p->ps->pages; i++)
719             ps_printf(p, "%zu 0 R", i * 4 +
720                    p->ps->pdfbody + 3);
721
722         base = (p->ps->pages - 1) * 4 +
723              p->ps->pdfbody + 4;
724
725         ps_printf(p, "]\n>>\nendobj\n");
726         pdf_obj(p, base);
727         ps_printf(p, "<<\n");
728         ps_printf(p, "/Type /Catalog\n");
729         ps_printf(p, "/Pages 2 0 R\n");
730         ps_printf(p, ">>\n");
731         xref = p->ps->pdfbytes;
732         ps_printf(p, "xref\n");
733         ps_printf(p, "0 %zu\n", base + 1);
734         ps_printf(p, "0000000000 65535 f\n");
735
736         for (i = 0; i < base; i++)
737             ps_printf(p, "%.10zu 00000 n\n",
738                    p->ps->pdfobjs[(int)i]);
739
740         ps_printf(p, "trailer\n");
741         ps_printf(p, "<<\n");
742         ps_printf(p, "/Size %zu\n", base + 1);
743         ps_printf(p, "/Root %zu 0 R\n", base);
744         ps_printf(p, "/Info 1 0 R\n");
745         ps_printf(p, ">>\n");
746         ps_printf(p, "startxref\n");
747         ps_printf(p, "%zu\n", xref);
748         ps_printf(p, "%%EOF\n");
749     }
750
751 static void
752 ps_begin(struct term *p)
753 {
754     time_t      t;
755     int         i;
756
757     /*
758     * Print margins into margin buffer. Nothing gets output to the
759     * screen yet, so we don't need to initialise the primary state.
760     */
761
762     if (p->ps->psmarg) {
763         assert(p->ps->psmargsz);
764         p->ps->psmarg[0] = '\0';
765     }
766
767     /*p->ps->pdfbytes = 0;*/
768     p->ps->psmargcur = 0;
769     p->ps->flags = PS_MARGINS;
770     p->ps->pscol = p->ps->left;
771     p->ps->psrow = p->ps->header;
772
773     ps_setfont(p, TERMFONT_NONE);
774
775     (*p->headf)(p, p->argf);
776     (*p->endline)(p);
777
778     p->ps->pscol = p->ps->left;
779     p->ps->psrow = p->ps->footer;
780
781     (*p->footf)(p, p->argf);

```



```

783     (*p->endline)(p);
785     p->ps->flags &= ~PS_MARGINS;
787     assert(0 == p->ps->flags);
788     assert(p->ps->psmarg);
789     assert('0' != p->ps->psmarg[0]);
791     /*
792     * Print header and initialise page state. Following this,
793     * stuff gets printed to the screen, so make sure we're sane.
794     */
796     t = time(NULL);
798     if (TERMTYPE_PS == p->type) {
799         ps_printf(p, "%!PS-Adobe-3.0\n");
800         ps_printf(p, "%%CreationDate: %s", ctime(&t));
801         ps_printf(p, "%%DocumentData: Clean7Bit\n");
802         ps_printf(p, "%%Orientation: Portrait\n");
803         ps_printf(p, "%%Pages: (atend)\n");
804         ps_printf(p, "%%PageOrder: Ascend\n");
805         ps_printf(p, "%%DocumentMedia: "
806             "Default %zu %zu 0 ( ) (\n",
807             (size_t)AFM2PNT(p, p->ps->width),
808             (size_t)AFM2PNT(p, p->ps->height));
809         ps_printf(p, "%%DocumentNeededResources: font");
811         for (i = 0; i < (int)TERMFONT_MAX; i++)
812             ps_printf(p, " %s", fonts[i].name);
814     } else {
815         ps_printf(p, "\n%%EndComments\n");
816         ps_printf(p, "%PDF-1.1\n");
817         pdf_obj(p, 1);
818         ps_printf(p, "<<\n");
819         ps_printf(p, ">>\n");
820         ps_printf(p, "endobj\n");
822         for (i = 0; i < (int)TERMFONT_MAX; i++) {
823             pdf_obj(p, (size_t)i + 3);
824             ps_printf(p, "<<\n");
825             ps_printf(p, "/Type /Font\n");
826             ps_printf(p, "/Subtype /Type1\n");
827             ps_printf(p, "/Name /F%zu\n", i);
828             ps_printf(p, "/BaseFont /%s\n", fonts[i].name);
829             ps_printf(p, ">>\n");
830         }
831     }
833     p->ps->pdfbody = (size_t)TERMFONT_MAX + 3;
834     p->ps->pscol = p->ps->left;
835     p->ps->psrow = p->ps->top;
836     p->ps->flags |= PS_NEWPAGE;
837     ps_setfont(p, TERMFONT_NONE);
838 }
841 static void
842 ps_pletter(struct term *p, int c)
843 {
844     int         f;
846     /*
847     * If we haven't opened a page context, then output that we're
848     * in a new page and make sure the font is correctly set.

```

```

849     */
851     if (PS_NEWPAGE & p->ps->flags) {
852         if (TERMTYPE_PS == p->type) {
853             ps_printf(p, "%%Page: %zu %zu\n",
854                 p->ps->pages + 1,
855                 p->ps->pages + 1);
856             ps_printf(p, "/%s %zu selectfont\n",
857                 fonts[(int)p->ps->lastf].name,
858                 p->ps->scale);
859         } else {
860             pdf_obj(p, p->ps->pdfbody +
861                 p->ps->pages * 4);
862             ps_printf(p, "<<\n");
863             ps_printf(p, "/Length %zu 0 R\n",
864                 p->ps->pdfbody + 1 +
865                 p->ps->pages * 4);
866             ps_printf(p, ">>\nstream\n");
867         }
868         p->ps->pdfastpg = p->ps->pdfbytes;
869         p->ps->flags &= ~PS_NEWPAGE;
870     }
871     /*
872     * If we're not in a PostScript "word" context, then open one
873     * now at the current cursor.
874     */
875     if (! (PS_INLINE & p->ps->flags)) {
876         if (TERMTYPE_PS != p->type) {
877             ps_printf(p, "BT\nF%d %zu Tf\n",
878                 (int)p->ps->lastf,
879                 p->ps->scale);
880             ps_printf(p, "%.3f %.3f Td\n",
881                 AFM2PNT(p, p->ps->pscol),
882                 AFM2PNT(p, p->ps->psrow));
883         } else {
884             ps_printf(p, "%.3f %.3f moveto\n",
885                 AFM2PNT(p, p->ps->pscol),
886                 AFM2PNT(p, p->ps->psrow));
887         }
888         p->ps->flags |= PS_INLINE;
889     }
890     assert(! (PS_NEWPAGE & p->ps->flags));
891     /*
892     * We need to escape these characters as per the PostScript
893     * specification. We would also escape non-graphable characters
894     * (like tabs), but none of them would get to this point and
895     * it's superfluous to abort() on them.
896     */
897     switch (c) {
898     case ('('):
899         /* FALLTHROUGH */
900     case (')'):
901         /* FALLTHROUGH */
902     case ('\\'):
903         ps_putchar(p, '\\');
904         break;
905     default:
906         break;
907     }
908     /* Write the character and adjust where we are on the page. */

```

```

915     f = (int)p->ps->lastf;
917     if (c <= 32 || (c - 32 >= MAXCHAR)) {
918         ps_putchar(p, ' ');
919         p->ps->pscol += (size_t)fonts[f].gly[0].wx;
920         return;
921     }
923     ps_putchar(p, (char)c);
924     c -= 32;
925     p->ps->pscol += (size_t)fonts[f].gly[c].wx;
926 }

929 static void
930 ps_pclose(struct term *p)
931 {
933     /*
934      * Spit out that we're exiting a word context (this is a
935      * "partial close" because we don't check the last-char buffer
936      * or anything).
937      */
939     if ( ! (PS_INLINE & p->ps->flags))
940         return;
942     if (TERMTYPE_PS != p->type) {
943         ps_printf(p, " Tj\nET\n");
944     } else
945         ps_printf(p, " show\n");
947     p->ps->flags &= ~PS_INLINE;
948 }

951 static void
952 ps_fclose(struct term *p)
953 {
955     /*
956      * Strong closure: if we have a last-char, spit it out after
957      * checking that we're in the right font mode. This will of
958      * course open a new scope, if applicable.
959      *
960      * Following this, close out any scope that's open.
961      */
963     if ('\0' != p->ps->last) {
964         if (p->ps->lastf != TERMFONT_NONE) {
965             ps_pclose(p);
966             ps_setfont(p, TERMFONT_NONE);
967         }
968         ps_pletter(p, p->ps->last);
969         p->ps->last = '\0';
970     }

972     if ( ! (PS_INLINE & p->ps->flags))
973         return;

975     ps_pclose(p);
976 }

979 static void
980 ps_letter(struct term *p, int arg)

```

```

981 {
982     char        cc, c;
984     /* LINTED */
985     c = arg >= 128 || arg <= 0 ? '?' : arg;
987     /*
988      * State machine dictates whether to buffer the last character
989      * or not. Basically, encoded words are detected by checking if
990      * we're an "8" and switching on the buffer. Then we put "8" in
991      * our buffer, and on the next character, flush both character
992      * and buffer. Thus, "regular" words are detected by having a
993      * regular character and a regular buffer character.
994      */
996     if ('\0' == p->ps->last) {
997         assert(8 != c);
998         p->ps->last = c;
999         return;
1000     } else if (8 == p->ps->last) {
1001         assert(8 != c);
1002         p->ps->last = '\0';
1003     } else if (8 == c) {
1004         assert(8 != p->ps->last);
1005         if ('_' == p->ps->last) {
1006             if (p->ps->lastf != TERMFONT_UNDER) {
1007                 ps_pclose(p);
1008                 ps_setfont(p, TERMFONT_UNDER);
1009             }
1010         } else if (p->ps->lastf != TERMFONT_BOLD) {
1011             ps_pclose(p);
1012             ps_setfont(p, TERMFONT_BOLD);
1013         }
1014         p->ps->last = c;
1015         return;
1016     } else {
1017         if (p->ps->lastf != TERMFONT_NONE) {
1018             ps_pclose(p);
1019             ps_setfont(p, TERMFONT_NONE);
1020         }
1021         cc = p->ps->last;
1022         p->ps->last = c;
1023         c = cc;
1024     }

1026     ps_pletter(p, c);
1027 }

1030 static void
1031 ps_advance(struct term *p, size_t len)
1032 {
1034     /*
1035      * Advance some spaces. This can probably be made smarter,
1036      * i.e., to have multiple space-separated words in the same
1037      * scope, but this is easier: just close out the current scope
1038      * and readjust our column settings.
1039      */
1041     ps_fclose(p);
1042     p->ps->pscol += len;
1043 }

1046 static void

```

```

1047 ps_endline(struct term *p)
1048 {
1050     /* Close out any scopes we have open: we're at eoln. */
1052     ps_fclose(p);
1054     /*
1055      * If we're in the margin, don't try to recalculate our current
1056      * row. XXX: if the column tries to be fancy with multiple
1057      * lines, we'll do nasty stuff.
1058      */
1060     if (PS_MARGINS & p->ps->flags)
1061         return;
1063     /* Left-justify. */
1065     p->ps->pscol = p->ps->left;
1067     /* If we haven't printed anything, return. */
1069     if (PS_NEWPAGE & p->ps->flags)
1070         return;
1072     /*
1073      * Put us down a line. If we're at the page bottom, spit out a
1074      * showpage and restart our row.
1075      */
1077     if (p->ps->psrow >= p->ps->lineheight +
1078         p->ps->bottom) {
1079         p->ps->psrow -= p->ps->lineheight;
1080         return;
1081     }
1083     ps_closepage(p);
1084 }
1087 static void
1088 ps_setfont(struct term *p, enum termfont f)
1089 {
1091     assert(f < TERMFONT_MAX);
1092     p->ps->lastf = f;
1093     /*
1094      * If we're still at the top of the page, let the font-setting
1095      * be delayed until we actually have stuff to print.
1096      */
1097     /*
1099     if (PS_NEWPAGE & p->ps->flags)
1100         return;
1102     if (TERMTYPE_PS == p->type)
1103         ps_printf(p, "%s %zu selectfont\n",
1104                 fonts[(int)f].name,
1105                 p->ps->scale);
1106     else
1107         ps_printf(p, "/F%d %zu Tf\n",
1108                 (int)f,
1109                 p->ps->scale);
1110 }

```

```

1113 /* ARGSUSED */
1114 static size_t
1115 ps_width(const struct term *p, int c)
1116 {
1118     if (c <= 32 || c - 32 >= MAXCHAR)
1119         return((size_t)fonts[(int)TERMFONT_NONE].gly[0].wx);
1121     c -= 32;
1122     return((size_t)fonts[(int)TERMFONT_NONE].gly[c].wx);
1123 }
1126 static double
1127 ps_hspan(const struct term *p, const struct roffsu *su)
1128 {
1129     double r;
1130     /*
1131      * All of these measurements are derived by converting from the
1132      * native measurement to AFM units.
1133      */
1134     switch (su->unit) {
1136     case (SCALE_CM):
1137         r = PNT2AFM(p, su->scale * 28.34);
1138         break;
1139     case (SCALE_IN):
1140         r = PNT2AFM(p, su->scale * 72);
1141         break;
1142     case (SCALE_PC):
1143         r = PNT2AFM(p, su->scale * 12);
1144         break;
1145     case (SCALE_PT):
1146         r = PNT2AFM(p, su->scale * 100);
1147         break;
1148     case (SCALE_EM):
1149         r = su->scale *
1150             fonts[(int)TERMFONT_NONE].gly[109 - 32].wx;
1151         break;
1152     case (SCALE_MM):
1153         r = PNT2AFM(p, su->scale * 2.834);
1154         break;
1155     case (SCALE_EN):
1156         r = su->scale *
1157             fonts[(int)TERMFONT_NONE].gly[110 - 32].wx;
1158         break;
1159     case (SCALE_VS):
1160         r = su->scale * p->ps->lineheight;
1161         break;
1162     default:
1163         r = su->scale;
1164         break;
1165     }
1166     return(r);
1168 }
1169 }
1171 static void
1172 ps_growbuf(struct term *p, size_t sz)
1173 {
1174     if (p->ps->psmargcur + sz <= p->ps->psmargsz)
1175         return;
1177     if (sz < PS_BUFSLOP)
1178         sz = PS_BUFSLOP;

```

```
1180     p->ps->psmargsz += sz;
1182     p->ps->psmarg = mandoc_realloc
1183         (p->ps->psmarg, p->ps->psmargsz);
1184 }
```

new/usr/src/cmd/mandoc/tree.c

1

```
6488 Sat Jul 19 19:24:33 2014
new/usr/src/cmd/mandoc/tree.c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
```

```
1 /* $Id: tree.c,v 1.47 2011/09/18 14:14:15 schwarze Exp $ */
2 /*
3  * Copyright (c) 2008, 2009, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <assert.h>
22 #include <limits.h>
23 #include <stdio.h>
24 #include <stdlib.h>
25 #include <time.h>
26
27 #include "mandoc.h"
28 #include "mdoc.h"
29 #include "man.h"
30 #include "main.h"
31
32 static void print_box(const struct eqn_box *, int);
33 static void print_man(const struct man_node *, int);
34 static void print_mdoc(const struct mdoc_node *, int);
35 static void print_span(const struct tbl_span *, int);
36
37
38 /* ARGSUSED */
39 void
40 tree_mdoc(void *arg, const struct mdoc *mdoc)
41 {
42
43     print_mdoc(mdoc_node(mdoc), 0);
44 }
45
46 /* ARGSUSED */
47 void
48 tree_man(void *arg, const struct man *man)
49 {
50 {
51
52     print_man(man_node(man), 0);
53 }
54
55
56 static void
```

new/usr/src/cmd/mandoc/tree.c

2

```
57 print_mdoc(const struct mdoc_node *n, int indent)
58 {
59     const char      *p, *t;
60     int             i, j;
61     size_t          argc, sz;
62     char            **params;
63     struct mdoc_argv *argv;
64
65     argv = NULL;
66     argc = sz = 0;
67     params = NULL;
68     t = p = NULL;
69
70     switch (n->type) {
71     case (MDOC_ROOT):
72         t = "root";
73         break;
74     case (MDOC_BLOCK):
75         t = "block";
76         break;
77     case (MDOC_HEAD):
78         t = "block-head";
79         break;
80     case (MDOC_BODY):
81         if (n->end)
82             t = "body-end";
83         else
84             t = "block-body";
85         break;
86     case (MDOC_TAIL):
87         t = "block-tail";
88         break;
89     case (MDOC_ELEM):
90         t = "elem";
91         break;
92     case (MDOC_TEXT):
93         t = "text";
94         break;
95     case (MDOC_TBL):
96         /* FALLTHROUGH */
97     case (MDOC_EQN):
98         break;
99     default:
100         abort();
101         /* NOTREACHED */
102     }
103
104     switch (n->type) {
105     case (MDOC_TEXT):
106         p = n->string;
107         break;
108     case (MDOC_BODY):
109         p = mdoc_macronames[n->tok];
110         break;
111     case (MDOC_HEAD):
112         p = mdoc_macronames[n->tok];
113         break;
114     case (MDOC_TAIL):
115         p = mdoc_macronames[n->tok];
116         break;
117     case (MDOC_ELEM):
118         p = mdoc_macronames[n->tok];
119         if (n->args) {
120             argv = n->args->argv;
121             argc = n->args->argc;
122         }
123     }
```

```

123     break;
124     case (MDOC_BLOCK):
125         p = mdoc_macronames[n->tok];
126         if (n->args) {
127             argv = n->args->argv;
128             argc = n->args->argc;
129         }
130         break;
131     case (MDOC_TBL):
132         /* FALLTHROUGH */
133     case (MDOC_EQN):
134         break;
135     case (MDOC_ROOT):
136         p = "root";
137         break;
138     default:
139         abort();
140         /* NOTREACHED */
141 }

143 if (n->span) {
144     assert(NULL == p && NULL == t);
145     print_span(n->span, indent);
146 } else if (n->eqn) {
147     assert(NULL == p && NULL == t);
148     print_box(n->eqn->root, indent);
149 } else {
150     for (i = 0; i < indent; i++)
151         putchar('\t');

153     printf("%s (%s)", p, t);

155     for (i = 0; i < (int)argc; i++) {
156         printf(" -%s", mdoc_argnames[argv[i].arg]);
157         if (argv[i].sz > 0)
158             printf(" [");
159         for (j = 0; j < (int)argv[i].sz; j++)
160             printf(" [%s]", argv[i].value[j]);
161         if (argv[i].sz > 0)
162             printf(" ]");
163     }

164     for (i = 0; i < (int)sz; i++)
165         printf(" [%s]", params[i]);

168     printf(" %d:%d\n", n->line, n->pos);
169 }

171 if (n->child)
172     print_mdcc(n->child, indent + 1);
173 if (n->next)
174     print_mdcc(n->next, indent);
175 }

178 static void
179 print_man(const struct man_node *n, int indent)
180 {
181     const char    *p, *t;
182     int           i;

184     t = p = NULL;

186     switch (n->type) {
187     case (MAN_ROOT):
188         t = "root";

```

```

189     break;
190     case (MAN_ELEM):
191         t = "elem";
192         break;
193     case (MAN_TEXT):
194         t = "text";
195         break;
196     case (MAN_BLOCK):
197         t = "block";
198         break;
199     case (MAN_HEAD):
200         t = "block-head";
201         break;
202     case (MAN_BODY):
203         t = "block-body";
204         break;
205     case (MAN_TAIL):
206         t = "block-tail";
207         break;
208     case (MAN_TBL):
209         /* FALLTHROUGH */
210     case (MAN_EQN):
211         break;
212     default:
213         abort();
214         /* NOTREACHED */
215 }

217 switch (n->type) {
218 case (MAN_TEXT):
219     p = n->string;
220     break;
221 case (MAN_ELEM):
222     /* FALLTHROUGH */
223 case (MAN_BLOCK):
224     /* FALLTHROUGH */
225 case (MAN_HEAD):
226     /* FALLTHROUGH */
227 case (MAN_TAIL):
228     /* FALLTHROUGH */
229 case (MAN_BODY):
230     p = man_macronames[n->tok];
231     break;
232 case (MAN_ROOT):
233     p = "root";
234     break;
235 case (MAN_TBL):
236     /* FALLTHROUGH */
237 case (MAN_EQN):
238     break;
239     default:
240         abort();
241         /* NOTREACHED */
242 }

244 if (n->span) {
245     assert(NULL == p && NULL == t);
246     print_span(n->span, indent);
247 } else if (n->eqn) {
248     assert(NULL == p && NULL == t);
249     print_box(n->eqn->root, indent);
250 } else {
251     for (i = 0; i < indent; i++)
252         putchar('\t');
253     printf("%s (%s) %d:%d\n", p, t, n->line, n->pos);
254 }

```

```

256     if (n->child)
257         print_man(n->child, indent + 1);
258     if (n->next)
259         print_man(n->next, indent);
260 }

262 static void
263 print_box(const struct eqn_box *ep, int indent)
264 {
265     int            i;
266     const char    *t;

268     if (NULL == ep)
269         return;
270     for (i = 0; i < indent; i++)
271         putchar('\t');

273     t = NULL;
274     switch (ep->type) {
275     case (EQN_ROOT):
276         t = "eqn-root";
277         break;
278     case (EQN_LIST):
279         t = "eqn-list";
280         break;
281     case (EQN_SUBEXPR):
282         t = "eqn-expr";
283         break;
284     case (EQN_TEXT):
285         t = "eqn-text";
286         break;
287     case (EQN_MATRIX):
288         t = "eqn-matrix";
289         break;
290     }

292     assert(t);
293     printf("%s(%d, %d, %d, %d, %d, \"%s\", \"%s\") %s\n",
294           t, EQN_DEFSIZE == ep->size ? 0 : ep->size,
295           ep->pos, ep->font, ep->mark, ep->pile,
296           ep->left ? ep->left : "",
297           ep->right ? ep->right : "",
298           ep->text ? ep->text : "");

300     print_box(ep->first, indent + 1);
301     print_box(ep->next, indent);
302 }

304 static void
305 print_span(const struct tbl_span *sp, int indent)
306 {
307     const struct tbl_dat *dp;
308     int            i;

310     for (i = 0; i < indent; i++)
311         putchar('\t');

313     switch (sp->pos) {
314     case (TBL_SPAN_HORIZ):
315         putchar('-');
316         return;
317     case (TBL_SPAN_DHORIZ):
318         putchar('=');
319         return;
320     default:

```

```

321         break;
322     }

324     for (dp = sp->first; dp; dp = dp->next) {
325         switch (dp->pos) {
326         case (TBL_DATA_HORIZ):
327             /* FALLTHROUGH */
328         case (TBL_DATA_NHORIZ):
329             putchar('-');
330             continue;
331         case (TBL_DATA_DHORIZ):
332             /* FALLTHROUGH */
333         case (TBL_DATA_NDHORIZ):
334             putchar('=');
335             continue;
336         default:
337             break;
338         }
339         printf("[\%s\"", dp->string ? dp->string : "");
340         if (dp->spans)
341             printf("(%d)", dp->spans);
342         if (NULL == dp->layout)
343             putchar(' ');
344         putchar(']');
345         putchar(' ');
346     }

348     printf("(tbl) %d:1\n", sp->line);
349 }

```

new/usr/src/cmd/mandoc/vol.c

1

1155 Sat Jul 19 19:24:33 2014

new/usr/src/cmd/mandoc/vol.c

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /*      $Id: vol.c,v 1.9 2011/03/22 14:33:05 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */
17 #ifdef HAVE_CONFIG_H
18 #include "config.h"
19 #endif
20
21 #include <stdlib.h>
22 #include <string.h>
23 #include <time.h>
24
25 #include "mdoc.h"
26 #include "mandoc.h"
27 #include "libmdoc.h"
28
29 #define LINE(x, y) \
30     if (0 == strcmp(p, x)) return(y);
31
32 const char *
33 mdoc_a2vol(const char *p)
34 {
35
36 #include "vol.in"
37
38     return(NULL);
39 }
```


new/usr/src/cmd/mandoc/vol.in

1

1408 Sat Jul 19 19:24:33 2014

new/usr/src/cmd/mandoc/vol.in

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 /*      $Id: vol.in,v 1.6 2010/06/19 20:46:28 kristaps Exp $ */
2 /*
3  * Copyright (c) 2009 Kristaps Dzonsons <kristaps@bsd.lv>
4  *
5  * Permission to use, copy, modify, and distribute this software for any
6  * purpose with or without fee is hereby granted, provided that the above
7  * copyright notice and this permission notice appear in all copies.
8  *
9  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
10 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
11 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
12 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
13 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
14 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
15 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
16 */

18 /*
19  * This file defines volume titles for .Dt.
20  *
21  * Be sure to escape strings.
22  */

24 LINE("USD",           "User\'s Supplementary Documents")
25 LINE("PS1",           "Programmer\'s Supplementary Documents")
26 LINE("AMD",           "Ancestral Manual Documents")
27 LINE("SMM",           "System Manager\'s Manual")
28 LINE("URM",           "User\'s Reference Manual")
29 LINE("PRM",           "Programmer\'s Manual")
30 LINE("KM",            "Kernel Manual")
31 LINE("IND",           "Manual Master Index")
32 LINE("MMI",           "Manual Master Index")
33 LINE("LOCAL",        "Local Manual")
34 LINE("LOC",           "Local Manual")
35 LINE("CON",           "Contributed Software Manual")
```

new/usr/src/man/Makefile

1

1815 Sat Jul 19 19:24:33 2014

new/usr/src/man/Makefile

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet
9 # at http://www.illumos.org/license/CDDL.
10 #
```

```
12 #
13 # Copyright 2011, Richard Lowe
14 # Copyright 2014 Nexenta Systems, Inc. All rights reserved.
14 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
15 #
```

```
17 SUBDIRS=      man1          \
18              man1b        \
19              man1c        \
20              man1has      \
21              man1m        \
22              man2         \
23              man3         \
24              man3bsm     \
25              man3c        \
26              man3c_db     \
27              man3cfgadm   \
28              man3commutil \
29              man3contract \
30              man3cpc      \
31              man3curses   \
32              man3dat      \
33              man3devid    \
34              man3devinfo  \
35              man3dlpi     \
36              man3dns_sd   \
37              man3elf      \
38              man3exacct   \
39              man3ext      \
40              man3fcoe     \
41              man3fstyp    \
42              man3gen      \
43              man3gss      \
44              man3head     \
45              man3iscsit   \
46              man3kstat    \
47              man3kvm      \
48              man3ldap     \
49              man3lgrp     \
50              man3lib      \
51              man3mail     \
52              man3malloc   \
53              man3mp       \
54              man3mpapi    \
55              man3nsl      \
```

new/usr/src/man/Makefile

2

```
56              man3nvpair  \
57              man3pam      \
58              man3papi     \
59              man3perl     \
60              man3picl     \
61              man3picltree \
62              man3pool     \
63              man3proc     \
64              man3project  \
65              man3resolv   \
66              man3rpc      \
67              man3rsm      \
68              man3sas1     \
69              man3scf      \
70              man3sec      \
71              man3secdb    \
72              man3sip      \
73              man3slp      \
74              man3socket   \
75              man3stmf     \
76              man3sysevent \
77              man3tecla    \
78              man3tnf      \
79              man3tsol     \
80              man3uuid     \
81              man3volmgt   \
82              man3xcurses  \
83              man3xnet     \
84              man4         \
85              man5         \
86              man7         \
87              man7d        \
88              man7fs       \
89              man7i        \
90              man7ipp      \
91              man7m        \
92              man7p        \
93              man9         \
94              man9e        \
95              man9f        \
96              man9p        \
97              man9s        \
```

```
99 .PARALLEL: $(SUBDIRS)
```

```
101 all           := TARGET = all
102 clean          := TARGET = clean
103 clobber        := TARGET = clobber
104 install        := TARGET = install
105 check          := TARGET = check
```

```
107 all check clean clobber install: $(SUBDIRS)
106 all clean clobber install: $(SUBDIRS)
```

```
109 $(SUBDIRS): FRC
110 @cd $@; pwd; $(MAKE) $(TARGET)
```

```
112 FRC:
```

```
*****
```

```
1290 Sat Jul 19 19:24:33 2014
```

```
new/usr/src/man/Makefile.man
```

```
5025 import and use mandoc
```

```
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
```

```
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
```

```
Reviewed by: Robert Mustacchi <rm@joyent.com>
```

```
Reviewed by: Albert Lee <trisk@nexenta.com>
```

```
Approved by: TBD
```

```
*****
```

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet
9 # at http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2011, Richard Lowe
14 # Copyright 2014 Nexenta Systems, Inc. All rights reserved.
14 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
15 #
16 #
17 MANDOC=      $(ONBLD_TOOLS)/bin/${MACH}/mandoc
18 ROOTMAN=     $(ROOT)/usr/share/man
19 ROOTHASMAN=  $(ROOT)/usr/has/man
20 FILEMODE=    0444
21 #
22 # The manual section being built, client Makefiles must set this to, for e.g.
23 # "3perl", with case matching that of the section name as installed.
24 #
25 # MANSECT=
26 #
27 MANCHECKS=    $(MANFILES:%=%.check)
28 ROOTMANFILES= $(MANFILES:%=$(ROOTMAN)/man$(MANSECT)/%)
29 ROOTMANLINKS= $(MANLINKS:%=$(ROOTMAN)/man$(MANSECT)/%)
30 #
31 $(ROOTMAN)/man$(MANSECT)/% $(ROOTHASMAN)/man$(MANSECT)/%: %
32     $(INS.file)
33 #
34 $(MANCHECKS):
35     @$$(ECHO) "checking $@:%.check="; \
36     $(MANDOC) -Tlint $@:%.check=
37 #
38 $(MANLINKS):
39     $(RM) $@; $(SYMLINK) $(LINKSRC) $@
40 #
41 $(ROOTMANLINKS): $(MANLINKS)
42     $(RM) $@; $(CP) -RP $(@F) $(@D)
43 #
44 all:
45 #
46 check:        $(MANCHECKS)
47 #
48 clean:
49 #
50 clobber:
51     $(RM) $(MANLINKS)
52 #
53 .PARALLEL:
```

```
55 FRC:
```

```

*****
13186 Sat Jul 19 19:24:33 2014
new/usr/src/man/man1/Makefile
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****

```

```

1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet
9 # at http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2011, Richard Lowe
14 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
15 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
16 #

```

18 include \$(SRC)/Makefile.master

20 MANSECT= 1

```

22 MANFILES= acctcom.1 //
23 adb.1 //
24 addbib.1 //
25 alias.1 //
26 allocate.1 //
27 amt.1 //
28 appcert.1 //
29 apptrace.1 //
30 apropos.1 //
31 ar.1 //
32 arch.1 //
33 asa.1 //
34 at.1 //
35 atq.1 //
36 atrm.1 //
37 audioconvert.1 //
38 audiocvt.1 //
39 audioplay.1 //
40 audiorecord.1 //
41 audiotest.1 //
42 auths.1 //
43 awk.1 //
44 banner.1 //
45 basename.1 //
46 bc.1 //
47 bdiff.1 //
48 bfs.1 //
49 break.1 //
50 builtin.1 //
51 cal.1 //
52 calendar.1 //
53 cancel.1 //
54 cat.1 //
55 cd.1 //
56 cdrw.1 //

```

```

57 checknr.1 //
58 chgrp.1 //
59 chkey.1 //
60 chmod.1 //
61 chown.1 //
62 ckdate.1 //
63 ckgid.1 //
64 ckint.1 //
65 ckitem.1 //
66 ckkeywd.1 //
67 ckpath.1 //
68 ckrange.1 //
69 ckstr.1 //
70 cksum.1 //
71 cktime.1 //
72 ckuid.1 //
73 ckyorn.1 //
74 clear.1 //
75 cmp.1 //
76 col.1 //
77 comm.1 //
78 command.1 //
79 compress.1 //
80 cp.1 //
81 cpio.1 //
82 cputrack.1 //
83 crle.1 //
84 crontab.1 //
85 crypt.1 //
86 csh.1 //
87 csplit.1 //
88 ctags.1 //
89 ctrun.1 //
90 ctstat.1 //
91 ctwatch.1 //
92 cut.1 //
93 date.1 //
94 dc.1 //
95 deallocate.1 //
96 deroff.1 //
97 dhcpinfo.1 //
98 diff.1 //
99 diff3.1 //
100 diffmk.1 //
101 digest.1 //
102 dircmp.1 //
103 dis.1 //
104 disown.1 //
105 dispuid.1 //
106 dispuid.1 //
107 dos2unix.1 //
108 download.1 //
109 dpost.1 //
110 du.1 //
111 dump.1 //
112 dumpcs.1 //
113 echo.1 //
114 ed.1 //
115 egrep.1 //
116 eject.1 //
117 elfdump.1 //
118 elfedit.1 //
119 elfsign.1 //
120 elfwrap.1 //
121 enable.1 //
122 encrypt.1 //

```

```

123          enhance.1
124          env.1
125          eqn.1
126          exec.1
127          exit.1
128          expand.1
129          expr.1
130          exstr.1
131          factor.1
132          fdformat.1
133          fgrep.1
134          file.1
135          filesync.1
136          find.1
137          finger.1
138          fmt.1
139          fmtmsg.1
140          fold.1
141          ftp.1
142          ftpcount.1
143          ftpwho.1
144          gcore.1
145          gencat.1
146          genmsg.1
147          getconf.1
148          getfacl.1
149          getlabel.1
150          getopt.1
151          getoptcvt.1
152          getopts.1
153          gettext.1
154          gettxt.1
155          getzonepath.1
156          glob.1
157          gprof.1
158          grep.1
159          groups.1
160          hash.1
161          head.1
162          history.1
163          hostid.1
164          hostname.1
165          iconv.1
166          indxbib.1
167          Intro.1
168          ipcrm.1
169          ipcs.1
170          isainfo.1
171          isalist.1
172          jobs.1
173          join.1
174          kbd.1
175          kdestroy.1
176          keylogin.1
177          keylogout.1
178          kill.1
179          kinit.1
180          klist.1
181          kmdb.1
182          kmfcfg.1
183          kpasswd.1
184          krb5-config.1
185          ksh93.1
186          ktutil.1
187          lari.1
188          last.1

```

```

189          lastcomm.1
190          ld.1
191          ldap.1
192          ldapdelete.1
193          ldaplist.1
194          ldapmodify.1
195          ldapmodrdn.1
196          ldapsearch.1
197          ldd.1
198          ld.so.1.1
199          let.1
200          lex.1
201          lgrpinfo.1
202          limit.1
203          line.1
204          list_devices.1
205          listusers.1
206          ln.1
207          loadkeys.1
208          locale.1
209          localedef.1
210          logger.1
211          login.1
212          logname.1
213          logout.1
214          look.1
215          lookbib.1
216          lorder.1
217          lp.1
218          lpstat.1
219          ls.1
220          m4.1
221          mac.1
222          mach.1
223          machid.1
224          madv.so.1.1
225          mail.1
226          mailcompat.1
227          mailq.1
228          mailstats.1
229          mailx.1
230          makekey.1
231          man.1
232          mandoc.1
233          mconnect.1
234          mcs.1
235          mdb.1
236          msg.1
237          mkdir.1
238          mkmsgs.1
239          mktemp.1
240          moe.1
241          more.1
242          mpss.so.1.1
243          msgcc.1
244          msgcpp.1
245          msgcvt.1
246          msgfmt.1
247          msggen.1
248          msgget.1
249          mt.1
250          mv.1
251          nawk.1
252          nc.1
253          nca.1
254          ncab2clf.1

```

```

255 ncakmod.1 //
256 newform.1 //
257 newgrp.1 //
258 news.1 //
259 newtask.1 //
260 nice.1 //
261 nl.1 //
262 nm.1 //
263 nohup.1 //
264 nroff.1 //
265 od.1 //
266 on.1 //
267 optisa.1 //
268 pack.1 //
269 pagesize.1 //
270 pargs.1 //
271 passwd.1 //
272 paste.1 //
273 pathchk.1 //
274 pax.1 //
275 pfexec.1 //
276 pg.1 //
277 pgregp.1 //
278 pkginfo.1 //
279 pkgmk.1 //
280 pkgparam.1 //
281 pkgproto.1 //
282 pkgtrans.1 //
283 pktool.1 //
284 plabel.1 //
285 plgrp.1 //
286 plimit.1 //
287 pmadvise.1 //
288 pmap.1 //
289 postio.1 //
290 postprint.1 //
291 postreverse.1 //
292 ppgsz.1 //
293 ppriv.1 //
294 pr.1 //
295 praliases.1 //
296 prctl.1 //
297 preap.1 //
298 prex.1 //
299 print.1 //
300 printf.1 //
301 priocntl.1 //
302 proc.1 //
303 prof.1 //
304 profiles.1 //
305 projects.1 //
306 ps.1 //
307 ptree.1 //
308 pvs.1 //
309 pwd.1 //
310 ranlib.1 //
311 rcapstat.1 //
312 rcp.1 //
313 rdist.1 //
314 read.1 //
315 readonly.1 //
316 refer.1 //
317 regcmp.1 //
318 renice.1 //
319 rev.1 //
320 rlogin.1 //

```

```

321 rm.1 //
322 rmformat.1 //
323 rmmount.1 //
324 roffbib.1 //
325 roles.1 //
326 rpcgen.1 //
327 rsh.1 //
328 runat.1 //
329 rup.1 //
330 ruptime.1 //
331 rusers.1 //
332 rwho.1 //
333 sar.1 //
334 scp.1 //
335 script.1 //
336 sdiff.1 //
337 sed.1 //
338 set.1 //
339 setfacl.1 //
340 setlabel.1 //
341 setpgpr.1 //
342 sftp.1 //
343 shcomp.1 //
344 shell_builtins.1 //
345 shift.1 //
346 size.1 //
347 sleep.1 //
348 smbutil.1 //
349 soelim.1 //
350 sort.1 //
351 sortbib.1 //
352 sotruss.1 //
353 spell.1 //
354 split.1 //
355 srchtxt.1 //
356 ssh.1 //
357 ssh-add.1 //
358 ssh-agent.1 //
359 ssh-http-proxy-connect.1 //
360 ssh-keygen.1 //
361 ssh-keyscan.1 //
362 ssh-socks5-proxy-connect.1 //
363 strchg.1 //
364 strings.1 //
365 strip.1 //
366 stty.1 //
367 sum.1 //
368 suspend.1 //
369 svcprop.1 //
370 svcs.1 //
371 symorder.1 //
372 sys-suspend.1 //
373 tabs.1 //
374 tail.1 //
375 talk.1 //
376 tar.1 //
377 tbl.1 //
378 tcopy.1 //
379 tee.1 //
380 telnet.1 //
381 test.1 //
382 tftp.1 //
383 time.1 //
384 times.1 //
385 timex.1 //
386 tip.1 //

```

```

387         tnfdump.1      \|
388         tnfxtract.1    \|
389         touch.1        \|
390         tput.1         \|
391         tr.1           \|
392         trap.1         \|
393         troff.1        \|
394         true.1         \|
395         truss.1        \|
396         tsort.1        \|
397         tty.1          \|
398         type.1         \|
399         typeset.1     \|
400         ul.1           \|
401         umask.1        \|
402         uname.1        \|
403         unifdef.1     \|
404         uniq.1         \|
405         units.1        \|
406         unix2dos.1    \|
407         uptime.1      \|
408         vacation.1    \|
409         vgrind.1       \|
410         volcheck.1    \|
411         volrmmount.1  \|
412         w.1           \|
413         wait.1         \|
414         wc.1           \|
413         what.1       \|
415         which.1       \|
416         who.1          \|
417         whocalls.1    \|
418         whois.1       \|
419         write.1        \|
420         xargs.1        \|
421         xgettext.1    \|
422         xstr.1         \|
423         yacc.1         \|
424         yes.1          \|
425         ypcat.1       \|
426         ypmatch.1     \|
427         yppasswd.1    \|
428         ypwhich.1     \|
429         zlogin.1      \|
430         zonename.1    \|

432 MANLINKS= batch.1    \|
433             bg.1       \|
434             case.1     \|
435             chdir.1    \|
436             checkeg.1  \|
437             continue.1 \|
438             decrypt.1  \|
439             dirname.1  \|
440             dirs.1     \|
441             disable.1  \|
442             dumpkeys.1 \|
443             edit.1     \|
444             errange.1  \|
445             errdate.1  \|
446             errgid.1   \|
447             errint.1   \|
448             erritem.1  \|
449             errpath.1  \|
450             errstr.1   \|
451             errtime.1  \|

```

```

452         erruid.1      \|
453         erryorn.1     \|
454         eval.1        \|
455         export.1      \|
456         false.1       \|
457         fc.1          \|
458         fg.1          \|
459         for.1         \|
460         foreach.1     \|
461         function.1    \|
462         goto.1        \|
463         hashcheck.1   \|
464         hashmake.1    \|
465         hashstat.1    \|
466         helpdate.1    \|
467         helpgid.1     \|
468         helpint.1     \|
469         helpitem.1    \|
470         helppath.1    \|
471         helprange.1   \|
472         helpstr.1     \|
473         helptime.1    \|
474         helpuid.1     \|
475         helpyorn.1    \|
476         hist.1        \|
477         i286.1        \|
478         i386.1        \|
479         i486.1        \|
480         i860.1        \|
481         iAPX286.1    \|
482         if.1          \|
483         intro.1       \|
484         jsh.1         \|
485         ksh.1         \|
486         ldapadd.1    \|
487         neqn.1        \|
488         notify.1      \|
489         onintr.1      \|
490         page.1        \|
491         pcat.1        \|
492         pcred.1       \|
493         pdpl1.1       \|
494         pfcsh.1       \|
495         pfiles.1      \|
496         pfksh.1       \|
497         pflags.1      \|
498         pfsh.1        \|
499         pkill.1       \|
500         pldd.1        \|
501         popd.1        \|
502         prun.1        \|
503         psig.1        \|
504         pstack.1      \|
505         pstop.1       \|
506         ptime.1       \|
507         pushd.1       \|
508         pwait.1       \|
509         pwdx.1        \|
510         red.1         \|
511         rehash.1      \|
512         remote_shell.1 \|
513         remsh.1       \|
514         repeat.1      \|
515         return.1      \|
516         rksh.1        \|
517         rksh93.1     \|

```

```

518      rmail.1      \|
519      rmdir.1     \|
520      rmumount.1  \|
521      select.1   \|
522      setenv.1   \|
523      settime.1  \|
524      sh.1       \|
525      snca.1     \|
526      source.1  \|
527      sparc.1   \|
528      spellin.1 \|
529      stop.1    \|
530      strconf.1 \|
531      sun.1     \|
532      switch.1  \|
533      u370.1   \|
534      u3b.1    \|
535      u3b15.1  \|
536      u3b2.1   \|
537      u3b5.1   \|
538      ulimit.1  \|
539      unalias.1 \|
540      uncompress.1 \|
541      unexpand.1 \|
542      unhash.1  \|
543      unlimit.1 \|
544      unpack.1  \|
545      unset.1   \|
546      unsetenv.1 \|
547      until.1   \|
548      valdate.1 \|
549      valgid.1  \|
550      valint.1  \|
551      valpath.1 \|
552      valrange.1 \|
553      valstr.1  \|
554      valtime.1 \|
555      valuid.1  \|
556      valyorn.1 \|
557      vax.1     \|
558      vedit.1   \|
559      whatis.1 \|
560      whence.1  \|
561      while.1   \|
562      zcat.1    \|

564 intro.1      := LINKSRC = Intro.1

566 whatis.1    := LINKSRC = apropos.1

568 unalias.1    := LINKSRC = alias.1

570 batch.1     := LINKSRC = at.1

572 dirname.1   := LINKSRC = basename.1

574 continue.1 := LINKSRC = break.1

576 chdir.1     := LINKSRC = cd.1
577 dirs.1      := LINKSRC = cd.1
578 popd.1      := LINKSRC = cd.1
579 pushd.1     := LINKSRC = cd.1

581 errdate.1   := LINKSRC = ckdate.1
582 helpdate.1 := LINKSRC = ckdate.1
583 valdate.1   := LINKSRC = ckdate.1

```

```

585 errgid.1    := LINKSRC = ckgid.1
586 helpgid.1   := LINKSRC = ckgid.1
587 valgid.1    := LINKSRC = ckgid.1

589 errint.1    := LINKSRC = ckint.1
590 helpint.1   := LINKSRC = ckint.1
591 valint.1    := LINKSRC = ckint.1

593 erritem.1   := LINKSRC = ckitem.1
594 helpitem.1  := LINKSRC = ckitem.1

596 errpath.1   := LINKSRC = ckpath.1
597 helppath.1  := LINKSRC = ckpath.1
598 valpath.1   := LINKSRC = ckpath.1

600 errrange.1  := LINKSRC = ckrange.1
601 helprange.1 := LINKSRC = ckrange.1
602 valrange.1  := LINKSRC = ckrange.1

604 errstr.1    := LINKSRC = ckstr.1
605 helpstr.1   := LINKSRC = ckstr.1
606 valstr.1    := LINKSRC = ckstr.1

608 errtime.1   := LINKSRC = cktime.1
609 helptime.1  := LINKSRC = cktime.1
610 valtime.1   := LINKSRC = cktime.1

612 erruid.1    := LINKSRC = ckuid.1
613 helpuid.1   := LINKSRC = ckuid.1
614 valuid.1    := LINKSRC = ckuid.1

616 erryorn.1   := LINKSRC = ckyorn.1
617 helpyorn.1 := LINKSRC = ckyorn.1
618 valyorn.1   := LINKSRC = ckyorn.1

620 uncompress.1 := LINKSRC = compress.1
621 zcat.1       := LINKSRC = compress.1

623 red.1       := LINKSRC = ed.1

625 disable.1   := LINKSRC = enable.1

627 decrypt.1  := LINKSRC = encrypt.1

629 checkeq.1   := LINKSRC = eqn.1
630 neqn.1      := LINKSRC = eqn.1

632 eval.1      := LINKSRC = exec.1
633 source.1    := LINKSRC = exec.1

635 goto.1      := LINKSRC = exit.1
636 return.1    := LINKSRC = exit.1

638 unexpand.1  := LINKSRC = expand.1

640 hashstat.1  := LINKSRC = hash.1
641 rehash.1    := LINKSRC = hash.1
642 unhash.1    := LINKSRC = hash.1

644 fc.1        := LINKSRC = history.1
645 hist.1      := LINKSRC = history.1

647 bg.1        := LINKSRC = jobs.1
648 fg.1        := LINKSRC = jobs.1
649 notify.1    := LINKSRC = jobs.1

```


new/usr/src/man/man1/Makefile

11

```

650 stop.1      := LINKSRC = jobs.1
652 jsh.1      := LINKSRC = ksh93.1
653 ksh.1      := LINKSRC = ksh93.1
654 rksh.1     := LINKSRC = ksh93.1
655 rksh93.1   := LINKSRC = ksh93.1
656 sh.1       := LINKSRC = ksh93.1

658 ldapadd.1  := LINKSRC = ldapmodify.1

660 ulimit.1   := LINKSRC = limit.1
661 unlimit.1  := LINKSRC = limit.1

663 dumpkeys.1 := LINKSRC = loadkeys.1

665 i286.1    := LINKSRC = machid.1
666 i386.1    := LINKSRC = machid.1
667 i486.1    := LINKSRC = machid.1
668 i860.1    := LINKSRC = machid.1
669 iAPX286.1 := LINKSRC = machid.1
670 pdp11.1   := LINKSRC = machid.1
671 sparc.1   := LINKSRC = machid.1
672 sun.1     := LINKSRC = machid.1
673 u370.1    := LINKSRC = machid.1
674 u3b.1     := LINKSRC = machid.1
675 u3b15.1   := LINKSRC = machid.1
676 u3b2.1    := LINKSRC = machid.1
677 u3b5.1    := LINKSRC = machid.1
678 vax.1     := LINKSRC = machid.1

680 rmail.1   := LINKSRC = mail.1

682 page.1   := LINKSRC = more.1

684 snca.1   := LINKSRC = nca.1

686 pcat.1   := LINKSRC = pack.1
687 unpack.1 := LINKSRC = pack.1

689 pfcsh.1  := LINKSRC = pfexec.1
690 pfksh.1  := LINKSRC = pfexec.1
691 pfsh.1   := LINKSRC = pfexec.1

693 pkill.1  := LINKSRC = pgrep.1

695 pcred.1  := LINKSRC = proc.1
696 pfiles.1 := LINKSRC = proc.1
697 pflags.1 := LINKSRC = proc.1
698 pldd.1   := LINKSRC = proc.1
699 prun.1   := LINKSRC = proc.1
700 psig.1   := LINKSRC = proc.1
701 pstack.1 := LINKSRC = proc.1
702 pstop.1  := LINKSRC = proc.1
703 ptime.1  := LINKSRC = proc.1
704 pwait.1  := LINKSRC = proc.1
705 pwdx.1   := LINKSRC = proc.1

707 rmdir.1  := LINKSRC = rm.1

709 rmumount.1 := LINKSRC = rmmount.1

711 remote_shell.1 := LINKSRC = rsh.1
712 remsh.1      := LINKSRC = rsh.1

714 export.1    := LINKSRC = set.1
715 setenv.1    := LINKSRC = set.1

```

new/usr/src/man/man1/Makefile

12

```

716 unset.1    := LINKSRC = set.1
717 unsetenv.1 := LINKSRC = set.1

719 case.1     := LINKSRC = shell_builtins.1
720 for.1       := LINKSRC = shell_builtins.1
721 foreach.1   := LINKSRC = shell_builtins.1
722 function.1  := LINKSRC = shell_builtins.1
723 if.1        := LINKSRC = shell_builtins.1
724 repeat.1   := LINKSRC = shell_builtins.1
725 select.1   := LINKSRC = shell_builtins.1
726 switch.1   := LINKSRC = shell_builtins.1
727 until.1    := LINKSRC = shell_builtins.1
728 while.1    := LINKSRC = shell_builtins.1

730 hashcheck.1 := LINKSRC = spell.1
731 hashmake.1  := LINKSRC = spell.1
732 spellin.1   := LINKSRC = spell.1

734 strconf.1  := LINKSRC = strchg.1

736 settime.1  := LINKSRC = touch.1

738 onintr.1   := LINKSRC = trap.1

740 false.1    := LINKSRC = true.1

742 whence.1   := LINKSRC = typeset.1

744 # Links to usr/has/man

746 edit.1     := LINKSRC = ../../../has/man/man1has/edit.1has
748 vedit.1    := LINKSRC = ../../../has/man/man1has/vi.1has

750 .KEEP_STATE:

752 include    $(SRC)/man/Makefile.man
754 install:   $(ROOTMANFILES) $(ROOTMANLINKS)

```

1745 Sat Jul 19 19:24:33 2014
 new/usr/src/man/man1/apropos.1
 5025 import and use mandoc
 Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
 Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
 Reviewed by: Robert Mustacchi <rm@joyent.com>
 Reviewed by: Albert Lee <trisk@nexenta.com>
 Approved by: TBD

```

1  \
2  \ This file and its contents are supplied under the terms of the
3  \ Common Development and Distribution License ("CDDL"), version 1.0.
4  \ You may only use this file in accordance with the terms of version
5  \ 1.0 of the CDDL.
6  \
7  \ A full copy of the text of the CDDL should have accompanied this
8  \ source. A copy of the CDDL is also available via the Internet at
9  \ http://www.illumos.org/license/CDDL.
10 \
11 \
12 \ Copyright 2012 Nexenta Systems, Inc. All rights reserved.
13 \ Copyright 2014 Garrett D'Amore <garrett@damore.org>
14 \
15 .Dd Jul 18, 2014
16 .Dt APROPOS 1
17 .Os
18 .Sh NAME
19 .Nm apropos ,
20 .Nm whatis
21 .Nd keyword search in
22 .Nm whatis
23 database files
24 .Sh SYNOPSIS
25 .Nm
26 .Op Fl M Ar path
27 .Op Fl s Ar section
28 .Ar keyword ...
29 .Nm whatis
30 .Op Fl M Ar path
31 .Op Fl s Ar section
32 .Ar keyword ...
33 .Sh DESCRIPTION
34 The
35 .Nm
36 utility searches a set of
37 .Nm whatis
38 database files matching each
39 .Ar keyword .
40 The
41 .Nm whatis
42 utility does the same search but only on complete words. The
43 .Nm whatis
44 database files are created using the
45 .Xr man 1
46 command.
47 .Sh OPTIONS
48 .Bl -tag -width ".Fl d"
49 .It Fl M Ar path
50 Force a specific colon separated manual path instead of the default search path.
51 Overrides the
52 .Ev MANPATH
53 environment variable.
54 .It Fl s Ar section
55 Restrict search to specified
56 .Ar section .

```

```

57 .El
58 .Sh ENVIRONMENT
59 The following environment variables affect the execution of
60 .Nm :
61 .Bl -tag -width ".Ev MANPATH , PATH"
62 .It Ev MANPATH , PATH
63 Used to find the location of the
64 .Nm whatis
65 database files.
66 .El
67 .Sh DIAGNOSTICS
68 The
69 .Nm
70 utility exits 0 if a keyword matched and 1 if no keywords are matched or no
71 .Nm whatis
72 databases are found.
73 .Sh INTERFACE STABILITY
74 .Nm Committed .
75 .Sh CODE SET INDEPENDENCE
76 Enabled.
77 .Sh SEE ALSO
78 .Xr man 1 ,
79 .Xr mandoc 1
80 \
81 \ Copyright (c) 1996, Sun Microsystems, Inc. All Rights Reserved
82 \ The contents of this file are subject to the terms of the Common Development
83 \ You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
84 \ When distributing Covered Code, include this CDDL HEADER in each file and in
85 .TH APROPOS 1 "Dec 20, 1996"
86 .SH NAME
87 apropos \- locate commands by keyword lookup
88 .SH SYNOPSIS
89 .LP
90 .LP
91 .nf
92 \fBapropos\fR \fRkeyword\fR...
93 .fi
94 \
95 .SH DESCRIPTION
96 .sp
97 .LP
98 The \fBapropos\fR utility displays the man page name, section number, and a
99 short description for each man page whose \fBNAME\fR line contains
100 \fRkeyword\fR. This information is contained in the \fB/usr/share/man/windex\fR
101 database created by \fBcatman\fR(1M). If \fBcatman\fR(1M) was not run, or was
102 run with the \fB-n\fR option, \fBapropos\fR fails. Each word is considered
103 separately and the case of letters is ignored. Words which are part of other
104 words are considered; for example, when looking for 'compile', \fBapropos\fR
105 finds all instances of 'compiler' also.
106 .sp
107 .LP
108 \fBapropos\fR is actually just the \fB-k\fR option to the \fBman\fR(1) command.
109 .SH EXAMPLES
110 .LP
111 \fBexample 1\fR To find a man page whose NAME line contains a keyword
112 .sp
113 .LP
114 Try
115 \
116 .sp
117 .in +2
118 .nf
119 example% \fBapropos password\fR
120 .fi
121 .in -2
122 .sp

```

```

44 .sp
45 .LP
46 and

48 .sp
49 .in +2
50 .nf
51 example% \fBapropos editor\fR
52 .fi
53 .in -2
54 .sp

56 .sp
57 .LP
58 If the line starts '\fIfilename\fR(\fIsection\fR) ./.\' you can run

60 .sp
61 .in +2
62 .nf
63 man -s \fIsection filename\fR
64 .fi
65 .in -2
66 .sp

68 .sp
69 .LP
70 to display the man page for \fIfilename\fR.

72 .LP
73 \fBExample 2\fR To find the man page for the subroutine \fBprintf()\fR
74 .sp
75 .LP
76 Try

78 .sp
79 .in +2
80 .nf
81 example% \fBapropos format\fR
82 .fi
83 .in -2
84 .sp

86 .sp
87 .LP
88 and then

90 .sp
91 .in +2
92 .nf
93 example% \fBman -s 3s printf\fR
94 .fi
95 .in -2
96 .sp

98 .sp
99 .LP
100 to get the manual page on the subroutine \fBprintf()\fR.

102 .SH FILES
103 .sp
104 .ne 2
105 .na
106 \fB\fB/usr/share/man/windex\fR \fR
107 .ad
108 .RS 26n
109 table of contents and keyword database

```

```

110 .RE

112 .SH ATTRIBUTES
113 .sp
114 .LP
115 See \fBattributes\fR(5) for descriptions of the following attributes:
116 .sp

118 .sp
119 .TS
120 box;
121 c | c
122 l | l .
123 ATTRIBUTE TYPE ATTRIBUTE VALUE
124 _
125 CSI Enabled
126 .TE

128 .SH SEE ALSO
129 .sp
130 .LP
131 \fBman\fR(1), \fBwhatism\fR(1), \fBcatman\fR(1M), \fBattributes\fR(5)
132 .SH DIAGNOSTICS
133 .sp
134 .ne 2
135 .na
136 \fB\f(CW/usr/share/man/windex: No such file or directory\fR\fR
137 .ad
138 .sp .6
139 .RS 4n
140 This database does not exist. \fBcatman\fR(1M) must be run to create it.
141 .RE

```

new/usr/src/man/man1/head.1

1

```
*****  
      8110 Sat Jul 19 19:24:33 2014  
new/usr/src/man/man1/head.1  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```

new/usr/src/man/man1/ld.1

1

```
*****  
59044 Sat Jul 19 19:24:33 2014  
new/usr/src/man/man1/ld.1  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```

9377 Sat Jul 19 19:24:34 2014

new/usr/src/man/man1/man.1

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1  \ " Copyright 2014 Garrett D'Amore <garrett@damore.org>
1  \ " te
2  \ " Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved.
3  \ " Copyright (c) 1980 Regents of the University of California.
4  \ " The Berkeley software License Agreement specifies the terms and conditions
5  \ " for redistribution.
6  .Dd Jul 18, 2014
7  .Dt MAN 1
8  .Os
9  .Sh NAME
10 .Nm man
11 .Nd find and display reference manual pages
12 .Sh SYNOPSIS
13 .Nm
14 .Op Fl
15 .Op Fl adFlrt
16 .Op Fl T Ar macro-package
17 .Op Fl M Ar path
18 .Op Fl s Ar section
19 .Ar name ...
20 .Nm
21 .Op Fl M Ar path
22 .Op Fl s Ar section
23 .Fl k
24 .Ar keyword
25 .Ar ...
26 .Nm
27 .Op Fl M Ar path
28 .Op Fl s Ar section
29 .Fl f
30 .Ar
31 .Nm
32 .Op Fl M Ar path
33 .Fl w
34 .Sh DESCRIPTION
35 The
36 .Nm
37 command displays information from the reference manuals. It
38 displays complete manual pages that you select by
39 .Ar name ,
40 or one-line summaries selected either by
41 .Ar keyword
42 .Pq Fl k ,
43 or by the name of an associated file
44 .Pq Fl f .
45 If no manual page is located,
46 .Nm
47 prints an error message.
48 .Ss "Source Format"
49 Reference Manual pages are marked up with either
50 .Xr man 5 ,
51 or
52 .Xr mdoc 5
53 language tags. The
54 .Nm
55 command recognizes the type of markup and

```

```

56 processes the file accordingly.
57 .
58 .Ss "Location of Manual Pages"
59 .
60 \ " Copyright (c) 1980 Regents of the University of California. The Berkeley sof
61 .TH MAN 1 "May 8, 2008"
62 .SH NAME
63 man \- find and display reference manual pages
64 .SH SYNOPSIS
65 .LP
66 .nf
67 \fBman\fR [\fB-\fR] [\fB-adFlrt\fR] [\fB-M\fR \fIpath\fR] [\fB-T\fR \fImacro-pac
68 .fi
69 .LP
70 .nf
71 \fBman\fR [\fB-M\fR \fIpath\fR] \fB-k\fR \fIkeyword\fR...
72 .fi
73 .LP
74 .nf
75 \fBman\fR [\fB-M\fR \fIpath\fR] \fB-f\fR \fIfile\fR...
76 .fi
77 .SH DESCRIPTION
78 .sp
79 .LP
80 The \fBman\fR command displays information from the reference manuals. It
81 displays complete manual pages that you select by \fIname\fR, or one-line
82 summaries selected either by \fIkeyword\fR (\fB-k\fR), or by the name of an
83 associated file (\fB-f\fR). If no manual page is located, \fBman\fR prints an
84 error message.
85 .Ss "Source Format"
86 .sp
87 .LP
88 Reference Manual pages are marked up with either \fBnroff\fR (see
89 \fBnroff\fR(1)) or \fBsgml\fR (Standard Generalized Markup Language) tags (see
90 \fBsgml\fR(5)). The \fBman\fR command recognizes the type of markup and
91 processes the file accordingly. The various source files are kept in separate
92 directories depending on the type of markup.
93 .Ss "Location of Manual Pages"
94 .sp
95 .LP
96 The online Reference Manual page directories are conventionally located in
97 .Pa /usr/share/man .
98 Each directory corresponds to a
99 \fB/usr/share/man/\fR. The \fBnroff\fR sources are located in the
100 \fB/usr/share/man/man/\fR* directories. The \fBsgml\fR sources are located in
101 the \fB/usr/share/man/sman/\fR* directories. Each directory corresponds to a
102 section of the manual. Since these directories are optionally installed, they
103 might not reside on your host. You might have to mount
104 .Pa /usr/share/man
105 might not reside on your host. You might have to mount \fB/usr/share/man/\fR
106 from a host on which they do reside.
107 The
108 .Nm
109 command reformats a page whenever it is requested.
110 .Pp
111 If the standard output is not a terminal, or if the
112 .Fl
113 flag is given,
114 .Nm
115 pipes its output through
116 .Xr cat 1 .
117 Otherwise,
118 .Nm

```

```

79 pipes its output through a pager such as
80 .Xr more 1
81 to handle paging and underlining on the screen.
82 .Sh OPTIONS
49 .sp
50 .LP
51 If there are preformatted, up-to-date versions in the corresponding \fBcat\fR*
52 or \fBfmt\fR* directories, \fBman\fR simply displays or prints those versions.
53 If the preformatted version of interest is out of date or missing, \fBman\fR
54 reformats it prior to display and stores the preformatted version if \fBcat\fR*
55 or \fBfmt\fR* is writable. The \fBwindex\fR database is not updated. See
56 \fBcatman\fR(1M). If directories for the preformatted versions are not
57 provided, \fBman\fR reformats a page whenever it is requested. \fBman\fR uses a
58 temporary file to store the formatted text during display.
59 .sp
60 .LP
61 If the standard output is not a terminal, or if the '\fB-\fR' flag is given,
62 \fBman\fR pipes its output through \fBcat\fR(1). Otherwise, \fBman\fR pipes its
63 output through \fBmore\fR(1) to handle paging and underlining on the screen.
64 .SH OPTIONS
65 .sp
66 .LP
83 The following options are supported:
84 .Bl -tag -width indent
85 .It Fl a
86 Shows all manual pages matching
87 .Ar name
88 within the
89 .Ev MANPATH
90 search path. Manual pages are displayed in the order found.
91 .It Fl d
68 .sp
69 .ne 2
70 .na
71 \fB\fB-a\fR\fR
72 .ad
73 .RS 20n
74 Shows all manual pages matching \fIname\fR within the \fBMANPATH\fR search
75 path. Manual pages are displayed in the order found.
76 .RE

78 .sp
79 .ne 2
80 .na
81 \fB\fB-d\fR\fR
82 .ad
83 .RS 20n
92 Debugs. Displays what a section-specifier evaluates to, method used for
93 searching, and paths searched by
94 .Nm .
95 .It Fl f Ar file ...
96 Attempts to locate manual pages related to any of the given
97 .Ar file
98 names. It strips the leading path name components from each
99 .Ar file ,
85 searching, and paths searched by \fBman\fR.
86 .RE

88 .sp
89 .ne 2
90 .na
91 \fB\fB-f\fR \fIfile ... \fR\fR
92 .ad
93 .RS 20n
94 \fBman\fR attempts to locate manual pages related to any of the given
95 \fIfile\fRs. It strips the leading path name components from each \fIfile\fR,

```

```

100 and then prints one-line summaries containing the resulting basename or names.
101 This option also uses the
102 .Pa whatis
103 database.
104 .It Fl F
105 This option is present for backwards compatibility and is documented
106 here for reference only. It performs no function.
107 .It Fl k Ar keyword ...
108 Prints out one-line summaries from the
109 .Pa whatis
110 database (table of contents) that contain any of the given
111 .Ar keyword .
112 The
113 .Pa whatis
114 database is created using the
115 .Fl w
116 option.
117 .It Fl l
118 Lists all manual pages found matching
119 .Ar name
120 within the search path.
121 .It Fl M Ar path
122 Specifies an alternate search path for manual pages. The
123 .Ar path
124 is a colon-separated list of directories that contain manual page directory
125 subtrees. For example, if
126 .Ar path
127 is
128 .Pa /usr/share/man:/usr/local/man ,
129 .Nm
130 searches for
131 .Ar name
132 in the standard location, and then
133 .Pa /usr/local/man .
134 When used with the
135 .Fl k ,
136 .Fl f ,
137 or
138 .Fl w
139 options, the
140 .Fl M
141 option must appear first. Each directory in the
142 .Ar path
143 is assumed to contain subdirectories of the form
144 .Pa man* ,
145 one for each section. This option overrides the
146 .Ev MANPATH
147 environment variable.
148 .It Fl r
149 Reformats the manual page, checking for formatting errors, but does not
150 display it.
151 .It Fl s Ar section
152 Specifies sections of the manual for
153 .Nm
154 to search. The directories searched for
155 .Ar name
156 are limited to those specified by
157 .Ar section .
158 .Ar section
159 can be a numerical digit, perhaps followed by one or more letters
160 to match the desired section of the manual, for example,
161 .Li "3libc".
162 Also,
163 .Ar section
164 can be a word, for example,
165 .Li local ,

```

```

166 .Li new ,
167 .Li old ,
168 .Li public .
169 .Ar section
170 can also be a letter. To specify multiple sections,
171 separate each section with a comma. This option overrides the
172 .Ev MANPATH
173 environment variable and the
174 .Pa man.cf
175 file. See
176 .Sx Search Path
177 below for an explanation of how
178 .Nm
179 conducts its search.
180 .It Fl t
181 Arranges for the specified manual pages to be sent to the default
182 printer as PostScript.
183 .It Fl T Ar macro-package
184 This option is present for backwards compatibility and is documented
185 here for reference only. It performs no function.
186 .It Fl w
187 Updates the
188 .Nm whatis
189 database.
190 .El
191 .Sh OPERANDS
192 This option also uses the \fBwindex\fR database.
193 .RE
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213 .sp
214 .ne 2
215 .na
216 \fB\FB-k\fR \fIkeyword ... \fR\fR
217 .ad
218 .RS 20n
219 Prints out one-line summaries from the \fBwindex\fR database (table of
220 contents) that contain any of the given \fIkeyword\fRs. The \fBwindex\fR
221 database is created using \fBcatman\fR(1M).
222 .RE
223
224
225
226
227
228
229
230
231
232
233
234
235
236 \fB\FB-M\fR \fIpath\fR\fR

```

```

237 .ad
238 .RS 20n
239 Specifies an alternate search path for manual pages. \fIpath\fR is a
240 colon-separated list of directories that contain manual page directory
241 subtrees. For example, if \fIpath\fR is \fB/usr/share/man:/usr/local/man\fR,
242 \fBman\fR searches for \fIname\fR in the standard location, and then
243 \fB/usr/local/man\fR. When used with the \fB-k\fR or \fB-f\fR options, the
244 \fB-M\fR option must appear first. Each directory in the \fIpath\fR is assumed
245 to contain subdirectories of the form \fBman\fR* or \fBsmman\fR* , one for each
246 section. This option overrides the \fBMANPATH\fR environment variable.
247 .RE
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```



```

192 The following operand is supported:
193 .Bl -tag -width indent
194 .It Ar name
204 .sp
205 .ne 2
206 .na
207 \fB\fIname\fR\fR
208 .ad
209 .RS 8n
195 The name of a standard utility or a keyword.
196 .El
197 .Sh USAGE
198 The usage of
199 .Nm
200 is described below:
201 .
202 .Ss "Manual Page Sections"
203 .
204 Entries in the reference manuals are organized into
205 .Em sections .
206 A section
211 .RE

213 .SH USAGE
214 .sp
215 .LP
216 The usage of \fBman\fR is described below:
217 .SS "Manual Page Sections"
218 .sp
219 .LP
220 Entries in the reference manuals are organized into \fIsection\fRs. A section
207 name consists of a major section name, typically a single digit, optionally
208 followed by a subsection name, typically one or more letters. An unadorned
209 major section name, for example,
210 .Qq 9 ,
211 does not act as an abbreviation for
212 the subsections of that name, such as
213 .Qq 9e ,
214 .Qq 9f ,
215 or
216 .Qq 9s .
217 That is, each subsection must be searched separately by
218 .Nm
219 .Fl s .
223 major section name, for example, "\fB9\fR", does not act as an abbreviation for
224 the subsections of that name, such as "\fB9e\fR", "\fB9f\fR", or "\fB9s\fR".
225 That is, each subsection must be searched separately by \fBman\fR \fB-s\fR.
220 Each section contains descriptions apropos to a particular reference category,
221 with subsections refining these distinctions. See the
222 .Em intro
223 manual pages for an explanation of the classification used in this release.
224 .
225 .Ss "Search Path"
226 .
227 Before searching for a given
228 .Ar name ,
229 .Nm
230 constructs a list of candidate directories and sections.
231 It searches for
232 .Ar name
233 in the directories specified by the
234 .Ev MANPATH
235 environment variable.
236 .Lp
237 In the absence of
238 .Ev MANPATH ,

```

```

239 .Nm
240 constructs its search path based upon the
241 .Ev PATH
242 environment variable, primarily by substituting
243 .Li man
244 for the last component of the
245 .Ev PATH
246 element. Special provisions are added
247 to account for unique characteristics of directories such as
248 .Pa /sbin ,
249 .Pa /usr/ucb ,
250 .Pa /usr/xpg4/bin ,
251 and others. If the file argument contains
252 a
253 .Qq /
254 character, the
255 .Em dirname
256 portion of the argument is used in place of
257 .Ev PATH
258 elements to construct the search path.
259 .Lp
260 Within the manual page directories,
261 .Nm
262 confines its search to the
227 with subsections refining these distinctions. See the \fBintro\fR manual pages
228 for an explanation of the classification used in this release.
229 .SS "Search Path"
230 .sp
231 .LP
232 Before searching for a given \fIname\fR, \fBman\fR constructs a list of
233 candidate directories and sections. \fBman\fR searches for \fIname\fR in the
234 directories specified by the \fBMANPATH\fR environment variable.
235 .sp
236 .LP
237 In the absence of \fBMANPATH\fR, \fBman\fR constructs its search path based
238 upon the \fBPATH\fR environment variable, primarily by substituting \fBman\fR
239 for the last component of the \fBPATH\fR element. Special provisions are added
240 to account for unique characteristics of directories such as \fB/sbin\fR,
241 \fB/usr/ucb\fR, \fB/usr/xpg4/bin\fR, and others. If the file argument contains
242 a \fB/\fR character, the \fBdirname\fR portion of the argument is used in place
243 of \fBPATH\fR elements to construct the search path.
244 .sp
245 .LP
246 Within the manual page directories, \fBman\fR confines its search to the
263 sections specified in the following order:
264 .Bl -bullet
265 .It
266 .Ar sections
267 specified on the command line with the
268 .Fl s
269 option
270 .It
271 .Ar sections
272 embedded in the
273 .Ev MANPATH
274 environment variable
275 .It
276 .Ar sections
277 specified in the
278 .Pa man.cf
279 file for each directory specified in the
280 .Ev MANPATH
281 environment variable
282 .El
283 If none of the above exist,
284 .Nm

```

```

285 searches each directory in the manual
286 .RS +4
287 .TP
288 .ie t \(\bu
289 .el o
290 \fIsection\fRs specified on the command line with the \fB-s\fR option
291 .RE
292 .RS +4
293 .TP
294 .ie t \(\bu
295 .el o
296 \fIsection\fRs embedded in the \fBMANPATH\fR environment variable
297 .RE
298 .RS +4
299 .TP
300 .ie t \(\bu
301 .el o
302 \fIsection\fRs specified in the \fBman.cf\fR file for each directory specified
303 in the \fBMANPATH\fR environment variable
304 .RE
305 .sp
306 .LP
307 If none of the above exist, \fBman\fR searches each directory in the manual
308 page path, and displays the first matching manual page found.
309 .Lp
310 The
311 .Pa man.cf
312 file has the following format:
313 .Lp
314 .Dl Pf MANSECTS= Ar section , Ns Op Ar section...
315 .Lp
316 Lines beginning with
317 .Sq Li #
318 and blank lines are considered comments, and are
319 ignored. Each directory specified in
320 .Ev MANPATH
321 can contain a manual page
322 .sp
323 .LP
324 The \fBman.cf\fR file has the following format:
325 .sp
326 .in +2
327 .nf
328 MANSECTS=\fIsection\fR[,\fIsection\fR]...
329 .fi
330 .in -2
331 .sp
332
333 .sp
334 .LP
335 Lines beginning with '\fB#\fR' and blank lines are considered comments, and are
336 ignored. Each directory specified in \fBMANPATH\fR can contain a manual page
337 configuration file, specifying the default search order for that directory.
338 .Sh "Referring to Other Manual Pages"
339 If the first line of the manual page is a reference to another manual
340 .SH FORMATTING MANUAL PAGES
341 .sp
342 .LP
343 Manual pages are marked up in \fBnroff\fR(1) or \fBsgml\fR(5). Nroff manual
344 pages are processed by \fBnroff\fR(1) or \fBtroff\fR(1) with the \fB-man\fR
345 macro package. Please refer to \fBman\fR(5) for information on macro usage.
346 \fBBSGML\fR(5) manual pages are processed by an \fBBSGML\fR parser and
347 passed to the formatter.
348 .SS "Preprocessing Nroff Manual Pages"
349 .sp
350 .LP

```

```

298 When formatting an nroff manual page, \fBman\fR examines the first line to
299 determine whether it requires special processing. If the first line is a string
300 of the form:
301 .sp
302 .in +2
303 .nf
304 \&'\"e\" \fIX\fR
305 .fi
306 .in -2
307 .sp
308
309 .sp
310 .LP
311 where \fIX\fR is separated from the '\fB'\fR' by a single SPACE and consists of
312 any combination of characters in the following list, \fBman\fR pipes its input
313 to \fBtroff\fR(1) or \fBnroff\fR(1) through the corresponding preprocessors.
314 .sp
315 .ne 2
316 .na
317 \fB\fBe\fR
318 .ad
319 .RS 5n
320 \fB\fBeqn\fR(1), or \fB\fBneqn\fR for \fBnroff\fR
321 .RE
322
323 .sp
324 .ne 2
325 .na
326 \fB\fBbr\fR
327 .ad
328 .RS 5n
329 \fB\fBrefer\fR(1)
330 .RE
331
332 .sp
333 .ne 2
334 .na
335 \fB\fBbt\fR
336 .ad
337 .RS 5n
338 \fB\fBtbl\fR(1)
339 .RE
340
341 .sp
342 .ne 2
343 .na
344 \fB\fBbv\fR
345 .ad
346 .RS 5n
347 \fB\fBvgrind\fR(1)
348 .RE
349
350 .sp
351 .LP
352 If \fB\fBeqn\fR or \fB\fBneqn\fR is invoked, it automatically reads the file
353 \fB/usr/pub/eqnchar\fR (see \fB\fBeqnchar\fR(5)). If \fBnroff\fR(1) is invoked,
354 \fB\fBcol\fR(1) is automatically used.
355 .SS "Referring to Other nroff Manual Pages"
356 .sp
357 .LP
358 If the first line of the nroff manual page is a reference to another manual
359 page entry fitting the pattern:
360 .Lp
361 .Dl \&.so man*/\fIsourcefile\fR
362 .Lp
363 .Nm

```

```

308 processes the indicated file in place of the current one. The
309 reference must be expressed as a path name relative to the root of the manual
310 page directory subtree.
311 .LP
312 When the second or any subsequent line starts with \fB&.so\fR, \fBman\fR
313 ignores it; \fBtroff\fR(1) or \fBnroff\fR(1) processes the request in the usual
314 manner.
315 .Sh ENVIRONMENT VARIABLES
316 See
317 .Xr environ 5
318 for descriptions of the following environment variables
319 that affect the execution of
320 .Nm man :
321 .Ev LANG ,
322 .Ev LC_ALL ,
323 .Ev LC_CTYPE ,
324 .Ev LC_MESSAGES ,
325 and
326 .Ev NLS_PATH .
327 .Bl -tag -width indent
328 .It Ev MANPATH
329 .SS "Processing SGML Manual Pages"
330 .sp
331 Manual pages are identified as being marked up in SGML by the presence of the
332 string \fB<!DOCTYPE\fR\&. If the file also contains the string
333 \fB$SHADOW_PAGE\fR, the file refers to another manual page for the content. The
334 reference is made with a file entity reference to the manual page that contains
335 the text. This is similar to the \fB&.so\fR mechanism used in the \fBnroff
336 formatted man pages.
337 .SH ENVIRONMENT VARIABLES
338 .sp
339 .LP
340 See \fBenviron\fR(5) for descriptions of the following environment variables
341 that affect the execution of \fBman\fR: \fBLC_ALL\fR, \fBLC_CTYPE\fR,
342 \fBLC_MESSAGES\fR, and \fBNLS_PATH\fR.
343 .sp
344 .ne 2
345 .na
346 \fB\fbmanpath\fR
347 .ad
348 .RS 11n
349 A colon-separated list of directories; each directory can be followed by a
350 comma-separated list of sections. If set, its value overrides
351 \fB/usr/share/man\fR as the default directory search path, and the \fBman.cf\fR
352 file as the default section search path. The \fB-M\fR and \fB-s\fR flags, in
353 turn, override these values.)
354 .It Ev PAGER
355 A program to use for interactively delivering
356 output to the screen. If not set,
357 .Sq Nm more Fl s
358 is used. See
359 .Xr more 1 .

```

```

340 .El
341 .Sh FILES
342 .Bl -tag -width indent
343 .It Pa /usr/share/man
344 .RE
345
346 .sp
347 .ne 2
348 .na
349 \fB\fbpager\fR
350 .ad
351 .RS 11n
352 A program to use for interactively delivering \fBman\fR's output to the screen.
353 If not set, '\fBmore\fR \fB-s\fR' is used. See \fBmore\fR(1).
354 .RE
355
356 .sp
357 .ne 2
358 .na
359 \fB\fbtcat\fR
360 .ad
361 .RS 11n
362 The name of the program to use to display \fBtroff\fR manual pages.
363 .RE
364
365 .sp
366 .ne 2
367 .na
368 \fB\fbtcat\fR
369 .ad
370 .RS 11n
371 The name of the formatter to use when the \fB-t\fR flag is given. If not set,
372 \fBtroff\fR(1) is used.
373 .RE
374
375 .SH EXAMPLES
376 .LP
377 \fBExample 1\fR \fRCreating a PostScript Version of a man page
378 .sp
379 .LP
380 The following example creates the \fBpipe\fR(2) man page in postscript for \fBcsh,
381 \fBtcsh, \fBksh and \fBsh users:
382 .sp
383 .in +2
384 .nf
385 % env TCAT=/usr/lib/lp/postscript/dpost man -t -s 2 pipe > pipe.ps
386 .fi
387 .in -2
388 .sp
389
390 .sp
391 .ne 2
392 .na
393 This is an alternative to using \fBman\fR \fB-t\fR, which sends the man page to
394 the default printer, if the user wants a postscript file version of the man
395 page.
396
397 .LP
398 \fBExample 2\fR \fRCreating a Text Version of a man page
399 .sp
400 .LP
401 The following example creates the \fBpipe\fR(2) man page in ascii text:
402 .sp
403 .in +2
404 .nf

```

```

466 man pipe.2 | col -x -b > pipe.text
467 .fi
468 .in -2
469 .sp

471 .sp
472 .LP
473 This is an alternative to using \fBman\fR \fB-t\fR, which sends the man page to
474 the default printer, if the user wants a text file version of the man page.

476 .SH EXIT STATUS
477 .sp
478 .LP
479 The following exit values are returned:
480 .sp
481 .ne 2
482 .na
483 \fB\fb0\fR\fR
484 .ad
485 .RS 6n
486 Successful completion.
487 .RE

489 .sp
490 .ne 2
491 .na
492 \fB\fb>0\fR\fR
493 .ad
494 .RS 6n
495 An error occurred.
496 .RE

498 .SH FILES
499 .sp
500 .ne 2
501 .na
502 \fB\fb/usr/share/man\fR\fR
503 .ad
504 .sp .6
505 .RS 4n
344 Root of the standard manual page directory subtree
345 .It Pa /usr/share/man/man?/*
346 Unformatted manual entries
347 .It Pa /usr/share/man/whatis
507 .RE

509 .sp
510 .ne 2
511 .na
512 \fB\fb/usr/share/man/man?/*\fR\fR
513 .ad
514 .sp .6
515 .RS 4n
516 Unformatted nroff manual entries
517 .RE

519 .sp
520 .ne 2
521 .na
522 \fB\fb/usr/share/man/sman?/*\fR\fR
523 .ad
524 .sp .6
525 .RS 4n
526 Unformatted \fB\fbSGML\fR manual entries
527 .RE

```

```

529 .sp
530 .ne 2
531 .na
532 \fB\fb/usr/share/man/cat?/*\fR\fR
533 .ad
534 .sp .6
535 .RS 4n
536 \fB\fbnroff\fR manual entries
537 .RE

539 .sp
540 .ne 2
541 .na
542 \fB\fb/usr/share/man/fmt?/*\fR\fR
543 .ad
544 .sp .6
545 .RS 4n
546 \fB\fbtroff\fR manual entries
547 .RE

549 .sp
550 .ne 2
551 .na
552 \fB\fb/usr/share/man/windex\fR\fR
553 .ad
554 .sp .6
555 .RS 4n
348 Table of contents and keyword database
349 .It Pa man.cf
557 .RE

559 .sp
560 .ne 2
561 .na
562 \fB\fb/usr/share/lib/tmac/an\fR\fR
563 .ad
564 .sp .6
565 .RS 4n
566 Standard \fB-man\fR macro package
567 .RE

569 .sp
570 .ne 2
571 .na
572 \fB\fb/usr/share/lib/sgml/locale/C/dtd/*\fR\fR
573 .ad
574 .sp .6
575 .RS 4n
576 \fB\fbSGML\fR document type definition files
577 .RE

579 .sp
580 .ne 2
581 .na
582 \fB\fb/usr/share/lib/sgml/locale/C/solbook/*\fR\fR
583 .ad
584 .sp .6
585 .RS 4n
586 \fB\fbSGML\fR style sheet and entity definitions directories
587 .RE

589 .sp
590 .ne 2
591 .na
592 \fB\fb/usr/share/lib/pub/eqnchar\fR\fR
593 .ad

```

```

594 .sp .6
595 .RS 4n
596 Standard definitions for \fBeqn\fR and \fBneqn\fR
597 .RE

599 .sp
600 .ne 2
601 .na
602 \fB\fBman.cf\fR\fR
603 .ad
604 .sp .6
605 .RS 4n
350 Default search order by section
351 .E1
352 .Sh EXIT STATUS
353 .Ex -std man
354 .Sh EXAMPLES
355 .
356 .Ss Example 1: Creating a PostScript Version of a man page
357 .
358 The following example spools the
359 .Xr pipe 2
360 man page in PostScript to the default printer:
361 .Pp
362 .Dl % man -t -s 2 pipe
363 .Pp
364 Note that
365 .Xr mandoc 1
366 can be used to obtain the PostScript content directly.
367 .Ss Example 2: Creating a Text Version of a man page
368 The following example creates the
369 .Xr pipe 2
370 man page in ASCII text:
371 .Pp
372 .Dl % man pipe.2 | col -x -b > pipe.text
373 .Sh CODE SET INDEPENDENCE
374 Enabled.
375 .Sh INTERFACE STABILITY
376 .Nm Committed .
377 .Sh SEE ALSO
378 .Xr apropos 1 ,
379 .Xr cat 1 ,
380 .Xr col 1 ,
381 .Xr mandoc 1 ,
382 .Xr more 1 ,
383 .Xr whatis 1 ,
384 .Xr environ 5 ,
385 .Xr man 5 ,
386 .Xr mdoc 5
387 .Sh NOTES
388 The
389 .Fl f
390 and
391 .Fl k
392 options use the
393 .Nm whatis
394 database, which is
395 created with the
396 .Fl w
397 option.
398 .Sh BUGS
607 .RE

609 .SH ATTRIBUTES
610 .sp
611 .LP

```

```

612 See \fBattributes\fR(5) for descriptions of the following attributes:
613 .sp

615 .sp
616 .TS
617 box;
618 c | c
619 l | l .
620 ATTRIBUTE TYPE ATTRIBUTE VALUE
621 _
622 CSI Enabled, see \fBNOTES\fR.
623 _
624 Interface Stability Committed
625 _
626 Standard See \fBstandards\fR(5).
627 .TE

629 .SH SEE ALSO
630 .sp
631 .LP
632 \fBapropos\fR(1), \fBcat\fR(1), \fBcol\fR(1), \fBdpost\fR(1), \fBeqn\fR(1),
633 \fBmore\fR(1), \fBnroff\fR(1), \fBrefer\fR(1), \fBtbl\fR(1), \fBtroff\fR(1),
634 \fBvgrind\fR(1), \fBwhatis\fR(1), \fBcatman\fR(1M), \fBattributes\fR(5),
635 \fBenviron\fR(5), \fBeqnchar\fR(5), \fBman\fR(5), \fBsgml\fR(5),
636 \fBstandards\fR(5)
637 .SH NOTES
638 .sp
639 .LP
640 The \fB-f\fR and \fB-k\fR options use the \fBwindex\fR database, which is
641 created by \fBcatman\fR(1M).
642 .sp
643 .LP
644 The \fBman\fR command is CSI-capable. However, some utilities invoked by the
645 \fBman\fR command, namely, \fBtroff\fR, \fBeqn\fR, \fBneqn\fR, \fBrefer\fR,
646 \fBtbl\fR, and \fBvgrind\fR, are not verified to be CSI-capable. Because of
647 this, the man command with the \fB-t\fR option can not handle non-EUC data.
648 Also, using the \fBman\fR command to display man pages that require special
649 processing through \fBeqn\fR, \fBneqn\fR, \fBrefer\fR, \fBtbl\fR, or
650 \fBvgrind\fR can not be CSI-capable.
651 .SH BUGS
652 .sp
653 .LP
399 The manual is supposed to be reproducible either on a phototypesetter or on an
400 ASCII terminal. However, on a terminal some information (indicated by
655 \fBASCII\fR terminal. However, on a terminal some information (indicated by
401 font changes, for instance) is lost.
657 .sp
658 .LP
659 Some dumb terminals cannot process the vertical motions produced by the \fBe\fR
660 (see \fBeqn\fR(1)) preprocessing flag. To prevent garbled output on these
661 terminals, when you use \fBe\fR, also use \fBt\fR, to invoke \fBcol\fR(1)
662 implicitly. This workaround has the disadvantage of eliminating superscripts
663 and subscripts, even on those terminals that can display them. Control-q clears
664 a terminal that gets confused by \fBeqn\fR(1) output.

```

14742 Sat Jul 19 19:24:34 2014

new/usr/src/man/man1/mandoc.1

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1  \
2  \
3  \
4  \
5  \
6  \
7  \
8  \
9  \
10 \
11 \
12 \
13 \
14 \
15 \
16 \
17 \
18 \
19 .Dd Jul 16, 2014
20 .Dt MANDOC 1
21 .Os
22 .Sh NAME
23 .Nm mandoc
24 .Nd format and display UNIX manuals
25 .Sh SYNOPSIS
26 .Nm mandoc
27 .Op Fl V
28 .Op Fl m Ns Ar format
29 .Op Fl O Ns Ar option
30 .Op Fl T Ns Ar output
31 .Op Fl W Ns Ar level
32 .Op Ar
33 .Sh DESCRIPTION
34 The
35 .Nm
36 utility formats
37 .Ux
38 manual pages for display.
39 .Pp
40 By default,
41 .Nm
42 reads
43 .Xr mdoc 5
44 or
45 .Xr man 5
46 text from stdin, implying
47 .Fl m Ns Cm andoc ,
48 and produces
49 .Fl T Ns Cm ascii
50 output.
51 .Pp
52 The arguments are as follows:
53 .Bl -tag -width Ds
54 .It Fl m Ns Ar format
55 Input format.
56 See

```

```

57 .Sx Input Formats
58 for available formats.
59 Defaults to
60 .Fl m Ns Cm andoc .
61 .It Fl O Ns Ar option
62 Comma-separated output options.
63 .It Fl T Ns Ar output
64 Output format.
65 See
66 .Sx Output Formats
67 for available formats.
68 Defaults to
69 .Fl T Ns Cm ascii .
70 .It Fl V
71 Print version and exit.
72 .It Fl W Ns Ar level
73 Specify the minimum message
74 .Ar level
75 to be reported on the standard error output and to affect the exit status.
76 The
77 .Ar level
78 can be
79 .Cm warning ,
80 .Cm error ,
81 or
82 .Cm fatal .
83 The default is
84 .Fl W Ns Cm fatal ;
85 .Fl W Ns Cm all
86 is an alias for
87 .Fl W Ns Cm warning .
88 See
89 .Sx EXIT STATUS
90 and
91 .Sx DIAGNOSTICS
92 for details.
93 .Pp
94 The special option
95 .Fl W Ns Cm stop
96 tells
97 .Nm
98 to exit after parsing a file that causes warnings or errors of at least
99 the requested level.
100 No formatted output will be produced from that file.
101 If both a
102 .Ar level
103 and
104 .Cm stop
105 are requested, they can be joined with a comma, for example
106 .Fl W Ns Cm error , Ns Cm stop .
107 .It Ar file
108 Read input from zero or more files.
109 If unspecified, reads from stdin.
110 If multiple files are specified,
111 .Nm
112 will halt with the first failed parse.
113 .El
114 .Ss Input Formats
115 The
116 .Nm
117 utility accepts
118 .Xr mdoc 5
119 and
120 .Xr man 5
121 input with
122 .Fl m Ns Cm doc

```

```

123 and
124 .Fl m Ns Cm an ,
125 respectively.
126 The
127 .Xr mdoc 5
128 format is
129 .Em strongly
130 recommended;
131 .Xr man 5
132 should only be used for legacy manuals.
133 .Pp
134 A third option,
135 .Fl m Ns Cm andoc ,
136 which is also the default, determines encoding on-the-fly: if the first
137 non-comment macro is
138 .Sq \&Dd
139 or
140 .Sq \&Dt ,
141 the
142 .Xr mdoc 5
143 parser is used; otherwise, the
144 .Xr man 5
145 parser is used.
146 .Pp
147 If multiple
148 files are specified with
149 .Fl m Ns Cm andoc ,
150 each has its file-type determined this way.
151 If multiple files are
152 specified and
153 .Fl m Ns Cm doc
154 or
155 .Fl m Ns Cm an
156 is specified, then this format is used exclusively.
157 .Ss Output Formats
158 The
159 .Nm
160 utility accepts the following
161 .Fl T
162 arguments, which correspond to output modes:
163 .Bl -tag -width "-Tlocale"
164 .It Fl T Ns Cm ascii
165 Produce 7-bit ASCII output.
166 This is the default.
167 See
168 .Sx ASCII Output .
169 .It Fl T Ns Cm html
170 Produce strict CSS1/HTML-4.01 output.
171 See
172 .Sx HTML Output .
173 .It Fl T Ns Cm lint
174 Parse only: produce no output.
175 Implies
176 .Fl W Ns Cm warning .
177 .It Fl T Ns Cm locale
178 Encode output using the current locale.
179 See
180 .Sx Locale Output .
181 .It Fl T Ns Cm man
182 Produce
183 .Xr man 5
184 format output.
185 See
186 .Sx Man Output .
187 .It Fl T Ns Cm pdf
188 Produce PDF output.

```

```

189 See
190 .Sx PDF Output .
191 .It Fl T Ns Cm ps
192 Produce PostScript output.
193 See
194 .Sx PostScript Output .
195 .It Fl T Ns Cm tree
196 Produce an indented parse tree.
197 .It Fl T Ns Cm utf8
198 Encode output in the UTF\ -8 multi-byte format.
199 See
200 .Sx UTF\ -8 Output .
201 .It Fl T Ns Cm xhtml
202 Produce strict CSS1/XHTML-1.0 output.
203 See
204 .Sx XHTML Output .
205 .El
206 .Pp
207 If multiple input files are specified, these will be processed by the
208 corresponding filter in-order.
209 .Ss ASCII Output
210 Output produced by
211 .Fl T Ns Cm ascii ,
212 which is the default, is rendered in standard 7-bit ASCII documented in
213 .Xr ascii 5 .
214 .Pp
215 Font styles are applied by using back-spaced encoding such that an
216 underlined character
217 .Sq c
218 is rendered as
219 .Sq _ Ns \e[bs] Ns c ,
220 where
221 .Sq \e[bs]
222 is the back-space character number 8.
223 Boldened characters are rendered as
224 .Sq c Ns \e[bs] Ns c .
225 .Pp
226 The special characters documented in
227 .Xr mandoc_char 5
228 are rendered best-effort in an ASCII equivalent.
229 If no equivalent is found,
230 .Sq \&?
231 is used instead.
232 .Pp
233 Output width is limited to 78 visible columns unless literal input lines
234 exceed this limit.
235 .Pp
236 The following
237 .Fl O
238 arguments are accepted:
239 .Bl -tag -width Ds
240 .It Cm indent Ns = Ns Ar indent
241 The left margin for normal text is set to
242 .Ar indent
243 blank characters instead of the default of five for
244 .Xr mdoc 5
245 and seven for
246 .Xr man 5 .
247 Increasing this is not recommended; it may result in degraded formatting,
248 for example overfull lines or ugly line breaks.
249 .It Cm width Ns = Ns Ar width
250 The output width is set to
251 .Ar width ,
252 which will normalise to \(>=60.
253 .El
254 .Ss HTML Output

```

```

255 Output produced by
256 .Fl T Ns Cm html
257 conforms to HTML-4.01 strict.
258 .Pp
259 The
260 .Pa example.style.css
261 file documents style-sheet classes available for customising output.
262 If a style-sheet is not specified with
263 .Fl O Ns Ar style ,
264 .Fl T Ns Cm html
265 defaults to simple output readable in any graphical or text-based web
266 browser.
267 .Pp
268 Special characters are rendered in decimal-encoded UTF\ -8.
269 .Pp
270 The following
271 .Fl O
272 arguments are accepted:
273 .Bl -tag -width Ds
274 .It Cm fragment
275 Omit the
276 .Aq !DOCTYPE
277 declaration and the
278 .Aq html ,
279 .Aq head ,
280 and
281 .Aq body
282 elements and only emit the subtree below the
283 .Aq body
284 element.
285 The
286 .Cm style
287 argument will be ignored.
288 This is useful when embedding manual content within existing documents.
289 .It Cm includes Ns = Ns Ar fmt
290 The string
291 .Ar fmt ,
292 for example,
293 .Ar ../src/%I.html ,
294 is used as a template for linked header files (usually via the
295 .Sq \&In
296 macro).
297 Instances of
298 .Sq \&I
299 are replaced with the include filename.
300 The default is not to present a
301 hyperlink.
302 .It Cm man Ns = Ns Ar fmt
303 The string
304 .Ar fmt ,
305 for example,
306 .Ar ../html%S/%N.%S.html ,
307 is used as a template for linked manuals (usually via the
308 .Sq \&Xr
309 macro).
310 Instances of
311 .Sq \&N
312 and
313 .Sq %S
314 are replaced with the linked manual's name and section, respectively.
315 If no section is included, section 1 is assumed.
316 The default is not to
317 present a hyperlink.
318 .It Cm style Ns = Ns Ar style.css
319 The file
320 .Ar style.css

```

```

321 is used for an external style-sheet.
322 This must be a valid absolute or
323 relative URI.
324 .El
325 .Ss Locale Output
326 Locale-dependent output encoding is triggered with
327 .Fl T Ns Cm locale .
328 This option is not available on all systems: systems without locale
329 support, or those whose internal representation is not natively UCS-4,
330 will fall back to
331 .Fl T Ns Cm ascii .
332 See
333 .Sx ASCII Output
334 for font style specification and available command-line arguments.
335 .Ss Man Output
336 Translate input format into
337 .Xr man 5
338 output format.
339 This is useful for distributing manual sources to legacy systems
340 lacking
341 .Xr mdoc 5
342 formatters.
343 .Pp
344 If
345 .Xr mdoc 5
346 is passed as input, it is translated into
347 .Xr man 5 .
348 If the input format is
349 .Xr man 5 ,
350 the input is copied to the output, expanding any
351 .Xr mandoc_roff 5
352 .Sq so
353 requests.
354 The parser is also run, and as usual, the
355 .Fl W
356 level controls which
357 .Sx DIAGNOSTICS
358 are displayed before copying the input to the output.
359 .Ss PDF Output
360 PDF-1.1 output may be generated by
361 .Fl T Ns Cm pdf .
362 See
363 .Sx PostScript Output
364 for
365 .Fl O
366 arguments and defaults.
367 .Ss PostScript Output
368 PostScript
369 .Qq Adobe-3.0
370 Level-2 pages may be generated by
371 .Fl T Ns Cm ps .
372 Output pages default to letter sized and are rendered in the Times font
373 family, 11-point.
374 Margins are calculated as 1/9 the page length and width.
375 Line-height is 1.4m.
376 .Pp
377 Special characters are rendered as in
378 .Sx ASCII Output .
379 .Pp
380 The following
381 .Fl O
382 arguments are accepted:
383 .Bl -tag -width Ds
384 .It Cm paper Ns = Ns Ar name
385 The paper size
386 .Ar name

```



```

387 may be one of
388 .Ar a3 ,
389 .Ar a4 ,
390 .Ar a5 ,
391 .Ar legal ,
392 or
393 .Ar letter .
394 You may also manually specify dimensions as
395 .Ar NNxNN ,
396 width by height in millimetres.
397 If an unknown value is encountered,
398 .Ar letter
399 is used.
400 .El
401 .Ss UTF\~8 Output
402 Use
403 .Fl T Ns Cm utf8
404 to force a UTF\~8 locale.
405 See
406 .Sx Locale Output
407 for details and options.
408 .Ss XHTML Output
409 Output produced by
410 .Fl T Ns Cm xhtml
411 conforms to XHTML-1.0 strict.
412 .Pp
413 See
414 .Sx HTML Output
415 for details; beyond generating XHTML tags instead of HTML tags, these
416 output modes are identical.
417 .Sh EXIT STATUS
418 The
419 .Nm
420 utility exits with one of the following values, controlled by the message
421 .Ar level
422 associated with the
423 .Fl W
424 option:
425 .Pp
426 .Bl -tag -width Ds -compact
427 .It 0
428 No warnings or errors occurred, or those that did were ignored because
429 they were lower than the requested
430 .Ar level .
431 .It 2
432 At least one warning occurred, but no error, and
433 .Fl W Ns Cm warning
434 was specified.
435 .It 3
436 At least one parsing error occurred, but no fatal error, and
437 .Fl W Ns Cm error
438 or
439 .Fl W Ns Cm warning
440 was specified.
441 .It 4
442 A fatal parsing error occurred.
443 .It 5
444 Invalid command line arguments were specified.
445 No input files have been read.
446 .It 6
447 An operating system error occurred, for example memory exhaustion or an
448 error accessing input files.
449 Such errors cause
450 .Nm
451 to exit at once, possibly in the middle of parsing or formatting a file.
452 .El

```

```

453 .Pp
454 Note that selecting
455 .Fl T Ns Cm lint
456 output mode implies
457 .Fl W Ns Cm warning .
458 .Sh EXAMPLES
459 To page manuals to the terminal:
460 .Pp
461 .Dl $ mandoc \-Wall,stop mandoc.1 2\*(Gt&1 | less
462 .Dl $ mandoc mandoc.1 mdoc.5 | less
463 .Pp
464 To produce HTML manuals with
465 .Ar style.css
466 as the style-sheet:
467 .Pp
468 .Dl $ mandoc \-Thtml -Ostyle=style.css mdoc.5 \*(Gt mdoc.5.html
469 .Pp
470 To check over a large set of manuals:
471 .Pp
472 .Dl $ mandoc \-Tlint `find /usr/src -name `e`e.[1-9]`
473 .Pp
474 To produce a series of PostScript manuals for A4 paper:
475 .Pp
476 .Dl $ mandoc \-Tps \-Opaper=a4 mdoc.5 man.5 \*(Gt manuals.ps
477 .Pp
478 Convert a modern
479 .Xr mdoc 5
480 manual to the older
481 .Xr man 5
482 format, for use on systems lacking an
483 .Xr mdoc 5
484 parser:
485 .Pp
486 .Dl $ mandoc \-Tman foo.mdoc \*(Gt foo.man
487 .Sh DIAGNOSTICS
488 Standard error messages reporting parsing errors are prefixed by
489 .Pp
490 .Sm off
491 .Dl Ar file : line : column : \ level :
492 .Sm on
493 .Pp
494 where the fields have the following meanings:
495 .Bl -tag -width "column"
496 .It Ar file
497 The name of the input file causing the message.
498 .It Ar line
499 The line number in that input file.
500 Line numbering starts at 1.
501 .It Ar column
502 The column number in that input file.
503 Column numbering starts at 1.
504 If the issue is caused by a word, the column number usually
505 points to the first character of the word.
506 .It Ar level
507 The message level, printed in capital letters.
508 .El
509 .Pp
510 Message levels have the following meanings:
511 .Bl -tag -width "warning"
512 .It Cm fatal
513 The parser is unable to parse a given input file at all.
514 No formatted output is produced from that input file.
515 .It Cm error
516 An input file contains syntax that cannot be safely interpreted,
517 either because it is invalid or because
518 .Nm

```

```

519 does not implement it yet.
520 By discarding part of the input or inserting missing tokens,
521 the parser is able to continue, and the error does not prevent
522 generation of formatted output, but typically, preparing that
523 output involves information loss, broken document structure
524 or unintended formatting.
525 .It Cm warning
526 An input file uses obsolete, discouraged or non-portable syntax.
527 All the same, the meaning of the input is unambiguous and a correct
528 rendering can be produced.
529 Documents causing warnings may render poorly when using other
530 formatting tools instead of
531 .Nm .
532 .El
533 .Pp
534 Messages of the
535 .Cm warning
536 and
537 .Cm error
538 levels are hidden unless their level, or a lower level, is requested using a
539 .Fl W
540 option or
541 .Fl T Ns Cm lint
542 output mode.
543 .Pp
544 The
545 .Nm
546 utility may also print messages related to invalid command line arguments
547 or operating system errors, for example when memory is exhausted or
548 input files cannot be read.
549 Such messages do not carry the prefix described above.
550 .Sh COMPATIBILITY
551 This section summarises
552 .Nm
553 compatibility with GNU troff.
554 Each input and output format is separately noted.
555 .Ss ASCII Compatibility
556 .Bl -bullet -compact
557 .It
558 Unrenderable unicode codepoints specified with
559 .Sq \e[uNNNN]
560 escapes are printed as
561 .Sq \&?
562 in mandoc.
563 In GNU troff, these raise an error.
564 .It
565 The
566 .Sq \&Bd \-literal
567 and
568 .Sq \&Bd \-unfilled
569 macros of
570 .Xr mdoc 5
571 in
572 .Fl T Ns Cm ascii
573 are synonyms, as are \-filled and \-ragged.
574 .It
575 In historic GNU troff, the
576 .Sq \&Pa
577 .Xr mdoc 5
578 macro does not underline when scoped under an
579 .Sq \&It
580 in the FILES section.
581 This behaves correctly in
582 .Nm .
583 .It
584 A list or display following the

```

```

585 .Sq \&Ss
586 .Xr mdoc 5
587 macro in
588 .Fl T Ns Cm ascii
589 does not assert a prior vertical break, just as it doesn't with
590 .Sq \&Sh .
591 .It
592 The
593 .Sq \&na
594 .Xr man 5
595 macro in
596 .Fl T Ns Cm ascii
597 has no effect.
598 .It
599 Words aren't hyphenated.
600 .El
601 .Ss HTML/XHTML Compatibility
602 .Bl -bullet -compact
603 .It
604 The
605 .Sq \efP
606 escape will revert the font to the previous
607 .Sq \ef
608 escape, not to the last rendered decoration, which is now dictated by
609 CSS instead of hard-coded.
610 It also will not span past the current scope,
611 for the same reason.
612 Note that in
613 .Sx ASCII Output
614 mode, this will work fine.
615 .It
616 The
617 .Xr mdoc 5
618 .Sq \&Bl \-hang
619 and
620 .Sq \&Bl \-tag
621 list types render similarly (no break following overreached left-hand
622 side) due to the expressive constraints of HTML.
623 .It
624 The
625 .Xr man 5
626 .Sq IP
627 and
628 .Sq TP
629 lists render similarly.
630 .El
631 .Sh INTERFACE STABILITY
632 The
633 .Nm
634 utility is
635 .Nm Committed ,
636 but the details of specific output formats other than ASCII are
637 .Nm Uncommitted .
638 .Sh SEE ALSO
639 .Xr eqn 5 ,
640 .Xr man 5 ,
641 .Xr mandoc_char 5 ,
642 .Xr mdoc 5 ,
643 .Xr mandoc_roff 5 ,
644 .Xr tbl 5
645 .Sh AUTHORS
646 The
647 .Nm
648 utility was written by
649 .An Kristaps Dzonsons ,
650 .Mt kristaps@bsd.lv .

```

```
651 .Sh CAVEATS
652 In
653 .Fl T Ns Cm html
654 and
655 .Fl T Ns Cm xhtml ,
656 the maximum size of an element attribute is determined by
657 .Dv BUFSIZ ,
658 which is usually 1024 bytes.
659 Be aware of this when setting long link
660 formats such as
661 .Fl O Ns Cm style Ns = Ns Ar really/long/link .
662 .Pp
663 Nesting elements within next-line element scopes of
664 .Fl m Ns Cm an ,
665 such as
666 .Sq br
667 within an empty
668 .Sq B ,
669 will confuse
670 .Fl T Ns Cm html
671 and
672 .Fl T Ns Cm xhtml
673 and cause them to forget the formatting of the prior next-line scope.
674 .Pp
675 The
676 .Sq \{aq
677 control character is an alias for the standard macro control character
678 and does not emit a line-break as stipulated in GNU troff.
```

new/usr/src/man/man1/printf.1

1

```
*****  
22738 Sat Jul 19 19:24:34 2014  
new/usr/src/man/man1/printf.1  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```

new/usr/src/man/man1/tar.1

1

```
*****  
34827 Sat Jul 19 19:24:34 2014  
new/usr/src/man/man1/tar.1  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_
```

1745 Sat Jul 19 19:24:34 2014
 new/usr/src/man/man1/whatis.1
 5025 import and use mandoc
 Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
 Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
 Reviewed by: Robert Mustacchi <rm@joyent.com>
 Reviewed by: Albert Lee <trisk@nexenta.com>
 Approved by: TBD

```

1  \
2  \ This file and its contents are supplied under the terms of the
3  \ Common Development and Distribution License ("CDDL"), version 1.0.
4  \ You may only use this file in accordance with the terms of version
5  \ 1.0 of the CDDL.
6  \
7  \ A full copy of the text of the CDDL should have accompanied this
8  \ source. A copy of the CDDL is also available via the Internet at
9  \ http://www.illumos.org/license/CDDL.
10 \
11 \
12 \ Copyright 2012 Nexenta Systems, Inc. All rights reserved.
13 \ Copyright 2014 Garrett D'Amore <garrett@damore.org>
14 \
15 .Dd Jul 18, 2014
16 .Dt APROPOS 1
17 .Os
18 .Sh NAME
19 .Nm apropos ,
20 .Nm whatis
21 .Nd keyword search in
22 .Nm whatis
23 database files
24 .Sh SYNOPSIS
25 .Nm
26 .Op Fl M Ar path
27 .Op Fl s Ar section
28 .Ar keyword ...
29 .Nm whatis
30 .Op Fl M Ar path
31 .Op Fl s Ar section
32 .Ar keyword ...
33 .Sh DESCRIPTION
34 The
35 .Nm
36 utility searches a set of
37 .Nm whatis
38 database files matching each
39 .Ar keyword .
40 The
41 .Nm whatis
42 utility does the same search but only on complete words. The
43 .Nm whatis
44 database files are created using the
45 .Xr man 1
46 command.
47 .Sh OPTIONS
48 .Bl -tag -width ".Fl d"
49 .It Fl M Ar path
50 Force a specific colon separated manual path instead of the default search path.
51 Overrides the
52 .Ev MANPATH
53 environment variable.
54 .It Fl s Ar section
55 Restrict search to specified
56 .Ar section .

```

```

57 .El
58 .Sh ENVIRONMENT
59 The following environment variables affect the execution of
60 .Nm :
61 .Bl -tag -width ".Ev MANPATH , PATH"
62 .It Ev MANPATH , PATH
63 Used to find the location of the
64 .Nm whatis
65 database files.
66 .El
67 .Sh DIAGNOSTICS
68 The
69 .Nm
70 utility exits 0 if a keyword matched and 1 if no keywords are matched or no
71 .Nm whatis
72 databases are found.
73 .Sh INTERFACE STABILITY
74 .Nm Committed .
75 .Sh CODE SET INDEPENDENCE
76 Enabled.
77 .Sh SEE ALSO
78 .Xr man 1 ,
79 .Xr mandoc 1
80 \
81 \ Copyright (c) 1992, Sun Microsystems, Inc.
82 \ The contents of this file are subject to the terms of the Common Development
83 \ You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
84 \ When distributing Covered Code, include this CDDL HEADER in each file and in
85 .TH WHATIS 1 "Sep 14, 1992"
86 .SH NAME
87 whatis \- display a one-line summary about a keyword
88 .SH SYNOPSIS
89 .LP
90 .nf
91 \fBwhatis\fR \fIcommand\fR...
92 .fi
93 .
94 .SH DESCRIPTION
95 .sp
96 .LP
97 \fBwhatis\fR looks up a given \fIcommand\fR and displays the header line from
98 the manual section. You can then run the \fBman\fR(1) command to get more
99 information. If the line starts \fBname(\fIsection\fR)\fR.\fR.\fR. you can do
100 \fBman\fR \fB-s\fR \fBsection\fR \fBname\fR to get the documentation for it.
101 Try \fBwhatis ed\fR and then you should do \fBman\fR \fB-s\fR \fBled\fR to get
102 the manual page for \fBed\fR(1).
103 .sp
104 .LP
105 \fBwhatis\fR is actually just the \fB-f\fR option to the \fBman\fR(1) command.
106 .sp
107 .LP
108 \fBwhatis\fR uses the \fB/usr/share/man/windex\fR database. This database is
109 created by \fBcatman\fR(1M). If this database does not exist, \fBwhatis\fR will
110 fail.
111 .SH FILES
112 .sp
113 .ne 2
114 .na
115 \fB/usr/share/man/windex\fR
116 .ad
117 .RS 25n
118 Table of contents and keyword database
119 .RE
120 .
121 .SH ATTRIBUTES
122 .sp

```

```
44 .LP
45 See \fBattributes\fR(5) for descriptions of the following attributes:
46 .sp

48 .sp
49 .TS
50 box;
51 c | c
52 l | l .
53 ATTRIBUTE TYPE ATTRIBUTE VALUE
54 -
55 CSI Enabled
56 .TE

58 .SH SEE ALSO
59 .sp
60 .LP
61 \fBapropos\fR(1), \fBman\fR(1), \fBcatman\fR(1M), \fBattributes\fR(5)
```

new/usr/src/man/man1m/arcstat.1m

1

```
*****  
4623 Sat Jul 19 19:24:34 2014  
new/usr/src/man/man1m/arcstat.1m  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```

1590 Sat Jul 19 19:24:35 2014

new/usr/src/man/man1m/catman.1m

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 .\"
2 .\" This file and its contents are supplied under the terms of the
3 .\" Common Development and Distribution License ("CDDL"), version 1.0.
4 .\" You may only use this file in accordance with the terms of version
5 .\" 1.0 of the CDDL.
6 .\"
7 .\" A full copy of the text of the CDDL should have accompanied this
8 .\" source. A copy of the CDDL is also available via the Internet at
9 .\" http://www.illumos.org/license/CDDL.
10 .\"
11 .\"
12 .\" Copyright 2014 Garrett D'Amore <garrett@damore.org>
13 .\"
14 .Dd Jul 19, 2014
15 .Dt CATMAN 1M
16 .Os
17 .Sh NAME
18 .Nm catman
19 .Nd generate
20 .Nm whatis
21 database files
22 .Sh SYNOPSIS
23 .Nm
24 .Op Fl M Ar path
25 .Op Fl w
26 .Sh DESCRIPTION
27 The
28 .Nm
29 utility generates a set of
30 .Nm whatis
31 database files suitable for use with
32 .Xr apropos 1
33 and
34 .Xr whatis 1 .
35 It is supplied for compatibility reasons. The same databases can
36 be generated using the
37 .Fl w
38 option with
39 .Xr man 1 ,
40 and that command should be used instead.
41 .Sh OPTIONS
42 .Bl -tag -width ".Fl d"
43 .It Fl M Ar path
44 Generate the
45 .Nm whatis
46 database files within the specified colon separated manual paths.
47 Overrides the
48 .Ev MANPATH
49 environment variable.
50 .It Fl w
51 This option is present for backwards compatibility, and is ignored.
52 .El
53 .Sh ENVIRONMENT
54 The following environment variables affect the execution of
55 .Nm :
56 .Bl -tag -width ".Ev MANPATH"
```

```

57 .It Ev MANPATH
58 Used to specify a colon separated list of manual paths within
59 which to generate
60 .Nm whatis
61 database files.
62 .El
63 .Sh EXIT STATUS
64 .Ex -std
65 .Sh INTERFACE STABILITY
66 .Nm "Obsolete Committed" .
67 .Sh CODE SET INDEPENDENCE
68 Enabled.
69 .Sh SEE ALSO
70 .Xr apropos 1 ,
71 .Xr man 1 ,
72 .Xr whatis 1
73 .\"
74 .\" Copyright (c) 1998 Sun Microsystems, Inc. All Rights Reserved.
75 .\" The contents of this file are subject to the terms of the Common Development
76 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
77 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
78 .TH CATMAN 1M "Feb 27, 1998"
79 .SH NAME
80 catman \- create the formatted files for the reference manual
81 .SH SYNOPSIS
82 .LP
83 .nf
84 \fBusr/bin/catman\fR [\fB-c\fR] [\fB-n\fR] [\fB-p\fR] [\fB-t\fR] [\fB-w\fR] [\f
85 [\fB-T\fR \fImacro-package\fR] [\fIsections\fR]
86 .fi
87 .SH DESCRIPTION
88 .sp
89 .LP
90 The \fBcatman\fR utility creates the preformatted versions of the on-line
91 manual from the \fBnroff\fR(1) or \fBsgml\fR(5) input files. This feature
92 allows easy distribution of the preformatted manual pages among a group of
93 associated machines (for example, with \fBrdist\fR(1)), since it makes the
94 directories of preformatted manual pages self-contained and independent of the
95 unformatted entries.
96 .sp
97 .LP
98 \fBcatman\fR also creates the \fBwindex\fR database file in the directories
99 specified by the \fBMANPATH\fR or the \fB-M\fR option. The \fBwindex\fR
100 database file is a three column list consisting of a keyword, the reference
101 page that the keyword points to, and a line of text that describes the purpose
102 of the utility or interface documented on the reference page. Each keyword is
103 taken from the comma separated list of words on the \fBNAME\fR line before the
104 '\(mi' (dash). The reference page that the keyword points to is the first word
105 on the \fBNAME\fR line. The text after the \(mi on the \fBNAME\fR line is the
106 descriptive text in the third column. The \fBNAME\fR line must be immediately
107 preceded by the page heading line created by the \fB&.TH\fR macro (see
108 \fBNOTES\fR for required format).
109 .sp
110 .LP
111 Each manual page is examined and those whose preformatted versions are missing
112 or out of date are recreated. If any changes are made, \fBcatman\fR recreates
113 the \fBwindex\fR database.
114 .sp
115 .LP
116 If a manual page is a \fBshadow\fR page, that is, it sources another manual
117 page for its contents, a symbolic link is made in the \fBcat\fR\fIix\fR or
118 \fBfmt\fR\fIix\fR directory to the appropriate preformatted manual page.
119 .sp
120 .LP
121 Shadow files in an unformatted nroff source file are identified by the first
```

```

51 line being of the form \fB&.so man\fR\fIx\fR\fB/yyy.\fR\fIx\fR\fB&.\fR
52 .sp
53 .LP
54 Shadow files in the \fBSGML\fR sources are identified by the string
55 \fBSHADOW_PAGE\fR. The file entity declared in the shadow file identifies the
56 file to be sourced.
57 .SH OPTIONS
58 .sp
59 .LP
60 The following options are supported:
61 .sp
62 .ne 2
63 .na
64 \fB\fB-c\fR\fR
65 .ad
66 .RS 20n
67 Create unformatted nroff source files in the appropriate \fBman\fR
68 subdirectories from the \fBSGML\fR sources. This option will overwrite any
69 existing file in the \fBman\fR directory of the same name as the \fBSGML\fR
70 file.
71 .RE

73 .sp
74 .ne 2
75 .na
76 \fB\fB-n\fR\fR
77 .ad
78 .RS 20n
79 Do not create (or recreate) the \fBwindex\fR database. If the \fB-n\fR option
80 is specified, the \fBwindex\fR database is not created and the \fBapropos\fR,
81 \fBwhatis\fR, \fBman\fR \fB-f\fR, and \fBman\fR \fB-k\fR commands will fail.
82 .RE

84 .sp
85 .ne 2
86 .na
87 \fB\fB-p\fR\fR
88 .ad
89 .RS 20n
90 Print what would be done instead of doing it.
91 .RE

93 .sp
94 .ne 2
95 .na
96 \fB\fB-t\fR\fR
97 .ad
98 .RS 20n
99 Create \fBtroff\fRed entries in the appropriate \fBfmt\fR subdirectories
100 instead of \fBnroff\fRing into the \fBcat\fR subdirectories.
101 .RE

103 .sp
104 .ne 2
105 .na
106 \fB\fB-w\fR\fR
107 .ad
108 .RS 20n
109 Only create the \fBwindex\fR database that is used by \fBwhatis\fR(1) and the
110 \fBman\fR(1) \fB-f\fR and \fB-k\fR options. No manual reformatting is done.
111 .RE

113 .sp
114 .ne 2
115 .na
116 \fB\fB-M\fR\fI directory\fR

```

```

117 .ad
118 .RS 20n
119 Update manual pages located in the specified \fIdirectory\fR,
120 (\fB/usr/share/man\fR by default). If the \fB-M\fR option is specified, the
121 directory argument must not contain a ',' (comma), since a comma is used to
122 delineate section numbers. See \fBman\fR(1).
123 .RE

125 .sp
126 .ne 2
127 .na
128 \fB\fB-T\fR\fI macro-package\fR\fR
129 .ad
130 .RS 20n
131 Use \fImacro-package\fR in place of the standard manual page macros, (
132 \fBman\fR(5) by default).
133 .RE

135 .SH OPERANDS
136 .sp
137 .LP
138 The following operand is supported:
139 .sp
140 .ne 2
141 .na
142 \fB\fIsections\fR\fR
143 .ad
144 .RS 12n
145 If there is one parameter not starting with a '\fB(mi\fR&', it is taken to be
146 a space separated list of manual sections to be processed by \fBcatman\fR. If
147 this operand is specified, only the manual sections in the list will be
148 processed. For example,
149 .sp
150 .in +2
151 .nf
152 \fBcatman 1 2 3\fR
153 .fi
154 .in -2
155 .sp

157 only updates manual sections \fB1\fR, \fB2\fR, and \fB3\fR. If specific
158 sections are not listed, all sections in the \fBman\fR directory specified by
159 the environment variable \fBMANPATH\fR are processed.
160 .RE

162 .SH ENVIRONMENT VARIABLES
163 .sp
164 .ne 2
165 .na
166 \fB\fBTROFF\fR\fR
167 .ad
168 .RS 11n
169 The name of the formatter to use when the \fB-t\fR flag is given. If not set,
170 \fBtroff\fR(1) is used.
171 .RE

173 .sp
174 .ne 2
175 .na
176 \fB\fBMANPATH\fR\fR
177 .ad
178 .RS 11n
179 A colon-separated list of directories that are processed by \fBcatman\fR and
180 \fBman\fR(1). Each directory can be followed by a comma-separated list of
181 sections. If set, its value overrides \fB/usr/share/man\fR as the default
182 directory search path, and the \fBman.cf\fR file as the default section search

```

```
183 path. The \fB-M\fR and \fB-s\fR flags, in turn, override these values.
184 .RE
```

```
186 .SH FILES
187 .sp
188 .ne 2
189 .na
190 \fB\fB/usr/share/man\fR\fR
191 .ad
192 .RS 28n
193 default manual directory location
194 .RE
```

```
196 .sp
197 .ne 2
198 .na
199 \fB\fB/usr/share/man/man*/.*\fR\fR
200 .ad
201 .RS 28n
202 raw nroff input files
203 .RE
```

```
205 .sp
206 .ne 2
207 .na
208 \fB\fB/usr/share/man/sman*/.*\fR\fR
209 .ad
210 .RS 28n
211 raw \fBBSGML\fR input files
212 .RE
```

```
214 .sp
215 .ne 2
216 .na
217 \fB\fB/usr/share/man/cat*/.*\fR\fR
218 .ad
219 .RS 28n
220 preformatted \fBnroff\fR manual pages
221 .RE
```

```
223 .sp
224 .ne 2
225 .na
226 \fB\fB/usr/share/man/fmt*/.*\fR\fR
227 .ad
228 .RS 28n
229 preformatted \fBtroff\fR manual pages
230 .RE
```

```
232 .sp
233 .ne 2
234 .na
235 \fB\fB/usr/share/man/windex\fR\fR
236 .ad
237 .RS 28n
238 table of contents and keyword database
239 .RE
```

```
241 .sp
242 .ne 2
243 .na
244 \fB\fB/usr/lib/makewhatis\fR\fR
245 .ad
246 .RS 28n
247 command script to make \fBwindex\fR database
248 .RE
```

```
250 .sp
251 .ne 2
252 .na
253 \fB\fB/usr/share/lib/tmac/an\fR\fR
254 .ad
255 .RS 28n
256 default macro package
257 .RE
```

```
259 .SH ATTRIBUTES
260 .sp
261 .LP
262 See \fBattributes\fR(5) for descriptions of the following attributes:
263 .sp
```

```
265 .sp
266 .TS
267 box;
268 c | c
269 l | l .
270 ATTRIBUTE TYPE ATTRIBUTE VALUE
271 _
272 CSI Enabled
273 .TE
```

```
275 .SH SEE ALSO
276 .sp
277 .LP
278 \fBbapropos\fR(1), \fBbman\fR(1), \fBbnroff\fR(1), \fBbrdist\fR(1), \fBbrm\fR(1),
279 \fBbtroff\fR(1), \fBbwhatis\fR(1), \fBbattributes\fR(5), \fBbman\fR(5),
280 \fBbsgm1\fR(5)
281 .SH DIAGNOSTICS
282 .sp
283 .ne 2
284 .na
285 \fB\fBman?/xxx.? (.so'ed from man?/yyy.?): No such file or directory\fR\fR
286 .ad
287 .sp .6
288 .RS 4n
289 The file outside the parentheses is missing, and is referred to by the file
290 inside them.
291 .RE
```

```
293 .sp
294 .ne 2
295 .na
296 \fB\fBtarget of .so in man?/xxx.? must be relative to /usr/man\fR\fR
297 .ad
298 .sp .6
299 .RS 4n
300 \fBbcatman\fR only allows references to filenames that are relative to the
301 directory \fB/usr/man\fR.
302 .RE
```

```
304 .sp
305 .ne 2
306 .na
307 \fB\fBopendir:man?:\fR \fBNo\fR \fBsuch\fR \fBfile\fR \fBor\fR
308 \fBdirectory\fR\fR
309 .ad
310 .sp .6
311 .RS 4n
312 A harmless warning message indicating that one of the directories \fBbcatman\fR
313 normally looks for is missing.
314 .RE
```

```
316 .sp
317 .ne 2
318 .na
319 \fB\fB*.*:\fR \fBNo\fR \fBsuch\fR \fBfile\fR \fBor\fR \fBdirectory\fR\fR
320 .ad
321 .sp .6
322 .RS 4n
323 A harmless warning message indicating \fBcatman\fR came across an empty
324 directory.
325 .RE

327 .SH WARNINGS
328 .sp
329 .LP
330 If a user, who has previously run \fBcatman\fR to install the \fBcat*\fR
331 directories, upgrades the operating system, the entire \fBcat*\fR directory
332 structure should be removed prior to running \fBcatman\fR. See \fBrm\fR(1).
333 .sp
334 .LP
335 Do not re-run \fBcatman\fR to re-build the \fBwhatis\fR database unless the
336 complete set of \fBman*\fR directories is present. \fBcatman\fR builds this
337 \fBwindex\fR file based on the \fBman*\fR directories.
338 .SH NOTES
339 .sp
340 .LP
341 To generate a valid windex index file, \fBcatman\fR has certain requirements.
342 Within the individual man page file, \fBcatman\fR requires two macro lines to
343 have a specific format. These are the \fB&.TH \fRpage heading line and the
344 \fB&.SH NAME \fRline.
345 .sp
346 .LP
347 The \fB&.TH \fRmacro requires at least the first three arguments, that is, the
348 filename, section number, and the date. The \fB&.TH \fRline starts off with
349 the \fB&.TH \fRmacro, followed by a space, the man page filename, a single
350 space, the section number, another single space, and the date. The date should
351 appear in double quotes and is specified as "day month year," with the month
352 always abbreviated to the first three letters (Jan, Feb, Mar, and so forth).
353 .sp
354 .LP
355 The \fB&.SH NAME \fRmacro, also known as the \fBNAME \fRline, must immediately
356 follow the \fB&.TH \fRline, with nothing in between those lines. No font
357 changes are permitted in the \fBNAME \fRline. The \fBNAME \fRline is
358 immediately followed by a line containing the man page filename; then shadow
359 page names, if applicable, separated by commas; a dash; and a brief summary
360 statement. These elements should all be on one line; no carriage returns are
361 permitted.
362 .sp
363 .LP
364 An example of proper coding of these lines is:
365 .sp
366 .in +2
367 .nf
368 \&.TH NISMATCH 1M "Apr "10, 1998"
369 \&.SH NAME
370 nismatch, nisgrep \e- utilities for searching NIS+ tables
371 .fi
372 .in -2
```

17974 Sat Jul 19 19:24:35 2014

new/usr/src/man/man1m/dd.1m

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 \" te
2 .\" Copyright (c) 2014, Joyent, Inc. All rights Reserved.
3 .\" Copyright (c) 1992, X/Open Company Limited All Rights Reserved
4 .\" Copyright 1989 AT&T
5 .\" Portions Copyright (c) 1995, Sun Microsystems, Inc. All Rights Reserved
6 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
7 .\" http://www.opengroup.org/bookstore/.
8 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
9 .\" This notice shall appear on any product containing this material.
10 .\" The contents of this file are subject to the terms of the Common Development
11 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
12 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
13 .TH DD 1M "Jan 04, 2014"
14 .SH NAME
15 dd \- convert and copy a file
16 .SH SYNOPSIS
17 .LP
18 .nf
19 \fB/usr/bin/dd\fR [\fIoperand=value\fR]...
20 .fi

22 .SH DESCRIPTION
23 .sp
24 .LP
25 The \fBdd\fR utility copies the specified input file to the specified output
26 with possible conversions. The standard input and output are used by default.
27 The input and output block sizes may be specified to take advantage of raw
28 physical I/O. Sizes are specified in bytes; a number may end with \fBk\fR,
29 \fBb\fR, or \fBw\fR to specify multiplication by 1024, 512, or 2, respectively.
30 Numbers may also be separated by \fBx\fR to indicate multiplication.
31 .sp
32 .LP
33 The \fBdd\fR utility reads the input one block at a time, using the specified
34 input block size. \fBdd\fR then processes the block of data actually returned,
35 which could be smaller than the requested block size. \fBdd\fR applies any
36 conversions that have been specified and writes the resulting data to the
37 output in blocks of the specified output block size.
38 .sp
39 .LP
40 \fBbcbs\fR is used only if \fBascii\fR, \fBasciib\fR, \fBbunblock\fR,
41 \fBebcdic\fR, \fBebcdicb\fR, \fBibm\fR, \fBibmb\fR, or \fBblock\fR conversion
42 is specified. In the first two cases, \fBcbs\fR characters are copied into the
43 conversion buffer, any specified character mapping is done, trailing blanks are
44 trimmed, and a \fBNEWLINE\fR is added before sending the line to output. In the
45 last three cases, characters up to \fBNEWLINE\fR are read into the conversion
46 buffer and blanks are added to make up an output record of size \fBcbs\fR.
47 \fBASCII\fR files are presumed to contain \fBNEWLINE\fR characters. If
48 \fBcbs\fR is unspecified or \fB0\fR, the \fBascii\fR, \fBasciib\fR,
49 \fBebcdic\fR, \fBebcdicb\fR, \fBibm\fR, and \fBibmb\fR options convert the
50 character set without changing the input file's block structure. The
51 \fBbunblock\fR and \fBblock\fR options become a simple file copy.
52 .sp
53 .LP
54 After completion, \fBdd\fR reports the number of whole and partial input and
55 output blocks.
56 .SH OPERANDS

```

```

57 .sp
58 .LP
59 The following operands are supported:
60 .sp
61 .ne 2
62 .na
63 \fB\bif=\fR\fIfile\fR\fR
64 .ad
65 .sp .6
66 .RS 4n
67 Specifies the input path. Standard input is the default.
68 .RE

70 .sp
71 .ne 2
72 .na
73 \fB\bof=\fR\fIfile\fR\fR
74 .ad
75 .sp .6
76 .RS 4n
77 Specifies the output path. Standard output is the default. If the
78 \fBseek=\fR\fIexpr\fR conversion is not also specified, the output file will be
79 truncated before the copy begins, unless \fBconv=notrunc\fR is specified. If
80 \fBseek=\fR\fIexpr\fR is specified, but \fBconv=notrunc\fR is not, the effect
81 of the copy will be to preserve the blocks in the output file over which
82 \fBdd\fR seeks, but no other portion of the output file will be preserved. (If
83 the size of the seek plus the size of the input file is less than the previous
84 size of the output file, the output file is shortened by the copy.)
85 .RE

87 .sp
88 .ne 2
89 .na
90 \fB\bibs=\fR\fIn\fR\fR
91 .ad
92 .sp .6
93 .RS 4n
94 Specifies the input block size in \fIn\fR bytes (default is \fB512\fR).
95 .RE

97 .sp
98 .ne 2
99 .na
100 \fB\bobs=\fR\fIn\fR\fR
101 .ad
102 .sp .6
103 .RS 4n
104 Specifies the output block size in \fIn\fR bytes (default is \fB512\fR).
105 .RE

107 .sp
108 .ne 2
109 .na
110 \fB\bbs=\fR\fIn\fR\fR
111 .ad
112 .sp .6
113 .RS 4n
114 Sets both input and output block sizes to \fIn\fR bytes, superseding \fBibs=\fR
115 and \fBobs=\fR. If no conversion other than \fBsync\fR, \fBnoerror\fR, and
116 \fBnotrunc\fR is specified, each input block is copied to the output as a
117 single block without aggregating short blocks.
118 .RE

120 .sp
121 .ne 2
122 .na

```

```

123 \fB\fBcbs=\fR\fIn\fR\fR
124 .ad
125 .sp .6
126 .RS 4n
127 Specifies the conversion block size for \fBblock\fR and \fBunblock\fR in bytes
128 by \fIn\fR (default is \fB0\fR). If \fBcbs=\fR is omitted or given a value of
129 \fB0\fR, using \fBblock\fR or \fBunblock\fR produces unspecified results.
130 .sp
131 This option is used only if \fBASCII\fR or \fBEBDIC\fR conversion is
132 specified. For the \fBascii\fR and \fBasciii\fR operands, the input is handled
133 as described for the \fBunblock\fR operand except that characters are converted
134 to \fBASCII\fR before the trailing \fBSPACE\fR characters are deleted. For the
135 \fBebdic\fR, \fBebdicb\fR, \fBibm\fR, and \fBibmb\fR operands, the input is
136 handled as described for the \fBblock\fR operand except that the characters are
137 converted to \fBEBDIC\fR or IBM \fBEBDIC\fR after the trailing \fBSPACE\fR
138 characters are added.
139 .RE

141 .sp
142 .ne 2
143 .na
144 \fB\fBfiles=\fR\fIn\fR\fR
145 .ad
146 .sp .6
147 .RS 4n
148 Copies and concatenates \fIn\fR input files before terminating (makes sense
149 only where input is a magnetic tape or similar device).
150 .RE

152 .sp
153 .ne 2
154 .na
155 \fB\fBskip=\fR\fIn\fR\fR
156 .ad
157 .sp .6
158 .RS 4n
159 Skips \fIn\fR input blocks (using the specified input block size) before
160 starting to copy. On seekable files, the implementation reads the blocks or
161 seeks past them. On non-seekable files, the blocks are read and the data is
162 discarded.
163 .RE

165 .sp
166 .ne 2
167 .na
168 \fB\fBiseek=\fR\fIn\fR\fR
169 .ad
170 .sp .6
171 .RS 4n
172 Seeks \fIn\fR blocks from beginning of input file before copying (appropriate
173 for disk files, where \fBskip\fR can be incredibly slow).
174 .RE

176 .sp
177 .ne 2
178 .na
179 \fB\fBoseek=\fR\fIn\fR\fR
180 .ad
181 .sp .6
182 .RS 4n
183 Seeks \fIn\fR blocks from beginning of output file before copying.
184 .RE

186 .sp
187 .ne 2
188 .na

```

```

189 \fB\fBseek=\fR\fIn\fR\fR
190 .ad
191 .sp .6
192 .RS 4n
193 Skips \fIn\fR blocks (using the specified output block size) from beginning of
194 output file before copying. On non-seekable files, existing blocks are read and
195 space from the current end-of-file to the specified offset, if any, is filled
196 with null bytes. On seekable files, the implementation seeks to the specified
197 offset or reads the blocks as described for non-seekable files.
198 .RE

200 .sp
201 .ne 2
202 .na
203 \fB\fBcount=\fR\fIn\fR\fR
204 .ad
205 .sp .6
206 .RS 4n
207 Copies only \fIn\fR input blocks.
208 .RE

210 .sp
211 .ne 2
212 .na
213 \fB\fBconv=\fR\fIvalue\fR[\fB,\fR\fIvalue\fR.\|.|\|]\fR
214 .ad
215 .sp .6
216 .RS 4n
217 Where \fIvalue\fRs are comma-separated symbols from the following list:
218 .sp
219 .ne 2
220 .na
221 \fB\fBascii\fR\fR
222 .ad
223 .RS 11n
224 Converts \fBEBDIC\fR to \fBASCII\fR.
225 .RE

227 .sp
228 .ne 2
229 .na
230 \fB\fBasciii\fR\fR
231 .ad
232 .RS 11n
233 Converts \fBEBDIC\fR to \fBASCII\fR using \fBBSD\fR-compatible character
234 translations.
235 .RE

237 .sp
238 .ne 2
239 .na
240 \fB\fBebdic\fR\fR
241 .ad
242 .RS 11n
243 Converts \fBASCII\fR to \fBEBDIC\fR. If converting fixed-length \fBASCII\fR
244 records without NEWLINES, sets up a pipeline with \fBdd conv=unblock\fR
245 beforehand.
246 .RE

248 .sp
249 .ne 2
250 .na
251 \fB\fBebdicb\fR\fR
252 .ad
253 .RS 11n
254 Converts \fBASCII\fR to \fBEBDIC\fR using \fBBSD\fR-compatible character

```

```

255 translations. If converting fixed-length \fBASCII\fR records without
256 \fBNEWLINE\fRs, sets up a pipeline with \fBdd conv=unblock\fR beforehand.
257 .RE

259 .sp
260 .ne 2
261 .na
262 \fB\fBibm\fR\fR
263 .ad
264 .RS 11n
265 Slightly different map of \fBASCII\fR to \fBEBCDIC\fR. If converting
266 fixed-length \fBASCII\fR records without \fBNEWLINE\fRs, sets up a pipeline
267 with \fBdd conv=unblock\fR beforehand.
268 .RE

270 .sp
271 .ne 2
272 .na
273 \fB\fBibmb\fR\fR
274 .ad
275 .RS 11n
276 Slightly different map of \fBASCII\fR to \fBEBCDIC\fR using
277 \fBBSD\fR-compatible character translations. If converting fixed-length
278 \fBASCII\fR records without \fBNEWLINE\fRs, sets up a pipeline with \fBdd
279 conv=unblock\fR beforehand.
280 .RE

282 The \fBascii\fR (or \fBasciib\fR), \fBebcdic\fR (or \fBebcdicb\fR), and
283 \fBibm\fR (or \fBibmb\fR) values are mutually exclusive.
284 .sp
285 .ne 2
286 .na
287 \fB\fBblock\fR\fR
288 .ad
289 .RS 11n
290 Treats the input as a sequence of \fBNEWLINE\fR-terminated or
291 \fBEOF\fR-terminated variable-length records independent of the input block
292 boundaries. Each record is converted to a record with a fixed length specified
293 by the conversion block size. Any \fBNEWLINE\fR character is removed from the
294 input line. \fBSPACE\fR characters are appended to lines that are shorter than
295 their conversion block size to fill the block. Lines that are longer than the
296 conversion block size are truncated to the largest number of characters that
297 will fit into that size. The number of truncated lines is reported.
298 .RE

300 .sp
301 .ne 2
302 .na
303 \fB\fBunblock\fR\fR
304 .ad
305 .RS 11n
306 Converts fixed-length records to variable length. Reads a number of bytes equal
307 to the conversion block size (or the number of bytes remaining in the input, if
308 less than the conversion block size), delete all trailing \fBSPACE\fR
309 characters, and append a \fBNEWLINE\fR character.
310 .RE

312 The \fBblock\fR and \fBunblock\fR values are mutually exclusive.
313 .sp
314 .ne 2
315 .na
316 \fB\fBlcase\fR\fR
317 .ad
318 .RS 9n
319 Maps upper-case characters specified by the \fBLC_CTYPE\fR keyword
320 \fBtolower\fR to the corresponding lower-case character. Characters for which

```

```

321 no mapping is specified are not modified by this conversion.
322 .RE

324 .sp
325 .ne 2
326 .na
327 \fB\fBucase\fR\fR
328 .ad
329 .RS 9n
330 Maps lower-case characters specified by the \fBLC_CTYPE\fR keyword
331 \fBtoupper\fR to the corresponding upper-case character. Characters for which
332 no mapping is specified are not modified by this conversion.
333 .RE

335 The \fBlcase\fR and \fBucase\fR symbols are mutually exclusive.
336 .sp
337 .ne 2
338 .na
339 \fB\fBswab\fR\fR
340 .ad
341 .RS 11n
342 Swaps every pair of input bytes. If the current input record is an odd number
343 of bytes, the last byte in the input record is ignored.
344 .RE

346 .sp
347 .ne 2
348 .na
349 \fB\fBnoerror\fR\fR
350 .ad
351 .RS 11n
352 Does not stop processing on an input error. When an input error occurs, a
353 diagnostic message is written on standard error, followed by the current input
354 and output block counts in the same format as used at completion. If the
355 \fBsync\fR conversion is specified, the missing input is replaced with null
356 bytes and processed normally. Otherwise, the input block will be omitted from
357 the output.
358 .RE

360 .sp
361 .ne 2
362 .na
363 \fB\fBnotrunc\fR\fR
364 .ad
365 .RS 11n
366 Does not truncate the output file. Preserves blocks in the output file not
367 explicitly written by this invocation of \fBdd\fR. (See also the preceding
368 \fBbof=\fR\fR operand.)
369 .RE

371 .sp
372 .ne 2
373 .na
374 \fB\fBsync\fR\fR
375 .ad
376 .RS 11n
377 Pads every input block to the size of the \fBbibs=\fR buffer, appending null
378 bytes. (If either \fBblock\fR or \fBunblock\fR is also specified, appends
379 \fBSPACE\fR characters, rather than null bytes.)
380 .RE

382 .RE

384 .sp
385 .ne 2
386 .na

```

```

387 \fB\fBoflag=\fR\fIvalue\fR[\fB,\fR\fIvalue\fR.\|.\|.\|]\fR
388 .ad
389 .sp 6
390 Where \fIvalue\fRs are comma-separated symbols from the following list which
391 affect the behavior of writing the output file:
392 .sp
393 .ne 2
394 .na
395 \fB\fBdsync\fR\fR
396 .ad
397 .RS 11n
398 The output file is opened with the \fBO_DSYNC\fR flag set. All data writes will
399 be synchronous. For more information on \fBO_DSYNC\fR see \fBfcntl.h\fR(3HEAD).
400 .RE

402 .sp
403 .ne 2
404 .na
405 \fB\fBsync\fR\fR
406 .ad
407 .RS 11n
408 The output file is opened with the \fBO_SYNC\fR flag set. All data and metadata
409 writes will be synchronous. For more information on \fBO_SYNC\fR see
410 \fBfcntl.h\fR(3HEAD).
411 .RE

413 .RE

413 .sp
414 .LP
415 If operands other than \fBconv=\fR and \fBoflag=\fR are specified more than once
416 the last specified \fBooperand=\fR\fIvalue\fR is used.
417 .sp
418 .LP
419 For the \fBbs=\fR, \fBcbs=\fR, \fBibs=\fR, and \fBobs=\fR operands, the
420 application must supply an expression specifying a size in bytes. The
421 expression, \fBexpr\fR, can be:
422 .RS +4
423 .TP
424 1.
425 a positive decimal number
426 .RE
427 .RS +4
428 .TP
429 2.
430 a positive decimal number followed by \fBk\fR, specifying multiplication by
431 1024
432 .RE
433 .RS +4
434 .TP
435 3.
436 a positive decimal number followed by \fBm\fR, specifying multiplication by
437 1024*1024
438 .RE
439 .RS +4
440 .TP
441 4.
442 a positive decimal number followed by \fBg\fR, specifying multiplication by
443 1024*1024*1024
444 .RE
445 .RS +4
446 .TP
447 5.
448 a positive decimal number followed by \fBt\fR, specifying multiplication by
449 1024*1024*1024*1024
450 .RE

```

```

451 .RS +4
452 .TP
453 6.
454 a positive decimal number followed by \fBP\fR, specifying multiplication by
455 1024*1024*1024*1024*1024
456 .RE
457 .RS +4
458 .TP
459 7.
460 a positive decimal number followed by \fBE\fR, specifying multiplication by
461 1024*1024*1024*1024*1024*1024
462 .RE
463 .RS +4
464 .TP
465 8.
466 a positive decimal number followed by \fBZ\fR, specifying multiplication by
467 1024*1024*1024*1024*1024*1024*1024
468 .RE
469 .RS +4
470 .TP
471 9.
472 a positive decimal number followed by \fBb\fR, specifying multiplication by
473 512
474 .RE
475 .RS +4
476 .TP
477 10.
478 two or more positive decimal numbers (with or without \fBk\fR or \fBb\fR)
479 separated by \fBx\fR, specifying the product of the indicated values.
480 .RE
481 .sp
482 .LP
483 All of the operands will be processed before any input is read.
484 .SH SIGNALS
485 .sp
486 .LP
487 When \fBdd\fR receives either SIGINFO or SIGUSR1, \fBdd\fR will emit the current
488 input and output block counts, total bytes written, total time elapsed, and the
489 number of bytes per second to standard error. This is the same information
490 format that \fBdd\fR emits when it successfully completes. Users may send
491 SIGINFO via their terminal. The default character is ^T, see \fBstty\fR(1) for
492 more information.
493 .sp
494 .LP
495 For \fBSIGINT\fR, \fBdd\fR writes status information to standard error before
496 exiting. \fBdd\fR takes the standard action for all other signals.

498 .SH USAGE
499 .sp
500 .LP
501 See \fBlargefile\fR(5) for the description of the behavior of \fBdd\fR when
502 encountering files greater than or equal to 2 Gbyte ( 2^31 bytes).
503 .SH EXAMPLES
504 .LP
505 \fBExample 1 \fRCopying from one tape drive to another
506 .sp
507 .LP
508 The following example copies from tape drive \fB0\fR to tape drive \fB1\fR,
509 using a common historical device naming convention.

511 .sp
512 .in +2
513 .nf
514 example% \fBdd if=/dev/rmt/0h of=/dev/rmt/1h\fR
515 .fi
516 .in -2

```



```

517 .sp
519 .LP
520 \fBExample 2 \fRStripping the first 10 bytes from standard input
521 .sp
522 .LP
523 The following example strips the first 10 bytes from standard input:
525 .sp
526 .in +2
527 .nf
528 example% \fBdd ibs=10 skip=1\fR
529 .fi
530 .in -2
531 .sp
533 .LP
534 \fBExample 3 \fRReading a tape into an ASCII file
535 .sp
536 .LP
537 This example reads an \fBEBCDIC\fR tape blocked ten 80-byte \fBEBCDIC\fR card
538 images per block into the \fBASCII\fR file \fBx\fR:
540 .sp
541 .in +2
542 .nf
543 example% \fBdd if=/dev/tape of=x ibs=800 cbs=80 conv=ascii,lcase\fR
544 .fi
545 .in -2
546 .sp
548 .LP
549 \fBExample 4 \fRUsing conv=sync to write to tape
550 .sp
551 .LP
552 The following example uses \fBconv=sync\fR when writing to a tape:
554 .sp
555 .in +2
556 .nf
557 example% \fBtar cvf - . | compress | dd obs=1024k of=/dev/rmt/0 conv=sync\fR
558 .fi
559 .in -2
560 .sp
562 .SH ENVIRONMENT VARIABLES
563 .sp
564 .LP
565 See \fBenvron\fR(5) for descriptions of the following environment variables
566 that affect the execution of \fBdd\fR: \fBBLANG\fR, \fBBLC_ALL\fR,
567 \fBBLC_CTYPE\fR, \fBBLC_MESSAGES\fR, and \fBNLSPATH\fR.
568 .SH EXIT STATUS
569 .sp
570 .LP
571 The following exit values are returned:
572 .sp
573 .ne 2
574 .na
575 \fB0\fR
576 .ad
577 .RS 6n
578 The input file was copied successfully.
579 .RE
581 .sp
582 .ne 2

```

```

583 .na
584 \fB>0\fR
585 .ad
586 .RS 6n
587 An error occurred.
588 .RE
590 .sp
591 .LP
592 If an input error is detected and the \fBnoerror\fR conversion has not been
593 specified, any partial output block will be written to the output file, a
594 diagnostic message will be written, and the copy operation will be
595 discontinued. If some other error is detected, a diagnostic message will be
596 written and the copy operation will be discontinued.
597 .SH ATTRIBUTES
598 .sp
599 .LP
600 See \fBattributes\fR(5) for descriptions of the following attributes:
601 .sp
603 .sp
604 .TS
605 box;
606 c | c
607 l | l .
608 ATTRIBUTE TYPE ATTRIBUTE VALUE
609 .
610 Interface Stability Standard
611 .TE
613 .SH SEE ALSO
614 .sp
615 .LP
616 \fBbcp\fR(1), \fBsed\fR(1), \fBtr\fR(1), \fBfcntl.h\fR(3HEAD),
617 \fBattributes\fR(5), \fBenvron\fR(5), \fBlargefile\fR(5), \fBstandards\fR(5)
618 .SH DIAGNOSTICS
619 .sp
620 .ne 2
621 .na
622 \fBfbf+p records in(out)\fR
623 .ad
624 .RS 23n
625 numbers of full and partial blocks read(written)
626 .RE
628 .SH NOTES
629 .sp
630 .LP
631 Do not use \fBdd\fR to copy files between file systems having different block
632 sizes.
633 .sp
634 .LP
635 Using a blocked device to copy a file will result in extra nulls being added
636 to the file to pad the final block to the block boundary.
637 .sp
638 .LP
639 When \fBdd\fR reads from a pipe, using the \fBibs=X\fR and \fBobs=Y\fR
640 operands, the output will always be blocked in chunks of size Y. When
641 \fBobs=Z\fR is used, the output blocks will be whatever was available to be read
642 from the pipe at the time.
643 .sp
644 .LP
645 When using \fBdd\fR to copy files to a tape device, the file size must be a
646 multiple of the device sector size (for example, 512 Kbyte). To copy files of
647 arbitrary size to a tape device, use \fBtar\fR(1) or \fBcpio\fR(1).

```

 22252 Sat Jul 19 19:24:35 2014
 new/usr/src/man/man1m/ipadm.1m
 5025 import and use mandoc
 Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
 Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
 Reviewed by: Robert Mustacchi <rm@joyent.com>
 Reviewed by: Albert Lee <trisk@nexenta.com>
 Approved by: TBD

```

1  \" te
2  .\" Copyright (c) 2012, Joyent, Inc. All Rights Reserved
3  .\" Copyright (c) 2013 by Delphix. All rights reserved.
4  .\" The contents of this file are subject to the terms of the Common Development
5  .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
6  .\" When distributing Covered Code, include this CDDL HEADER in each file and in
7  .TH IPADM 1M \"May 14, 2012\"
8  .SH NAME
9  ipadm \- configure IP network interfaces and protocol properties.
10 .SH SYNOPSIS
11 .LP
12 .nf
13 \fBipadm\fR create-if [\fB-t\fR] \fIinterface\fR
14 .fi

16 .LP
17 .nf
18 \fBipadm\fR disable-if [\fB-t\fR] \fIinterface\fR
19 .fi

21 .LP
22 .nf
23 \fBipadm\fR enable-if [\fB-t\fR] \fIinterface\fR
24 .fi

26 .LP
27 .nf
28 \fBipadm\fR delete-if \fIinterface\fR
29 .fi

31 .LP
32 .nf
33 \fBipadm\fR show-if [[\fB-p\fR] \fB-o\fR \fIfield\fR[,...]] [\fIinterface\fR]
34 .fi

36 .LP
37 .nf
38 \fBipadm\fR set-ifprop [\fB-t\fR] \fB-p\fR \fIprop\fR=<\fIvalue\fR[,...]> \fB-m\
39 .fi

41 .LP
42 .nf
43 \fBipadm\fR reset-ifprop [\fB-t\fR] \fB-p\fR \fIprop\fR \fB-m\fR \fIprotocol\fR
44 .fi

46 .LP
47 .nf
48 \fBipadm\fR show-ifprop [[\fB-c\fR]\fB-o\fR \fIfield\fR[,...]] [\fB-p\fR \fIprop
49 [\fIinterface\fR]
50 .fi

52 .LP
53 .nf
54 \fBipadm\fR create-addr [\fB-t\fR] \fB-T\fR static [\fB-d\fR]
55 \fB-a\fR {local|remote}=\fIaddr\fR[/\fIprefixlen\fR],... \fIaddrobj\fR
56 .fi

```

```

58 .LP
59 .nf
60 \fBipadm\fR create-addr [\fB-t\fR] \fB-T\fR dhcp [\fB-w\fR \fIseconds\fR | forev
61 .fi

63 .LP
64 .nf
65 \fBipadm\fR create-addr [\fB-t\fR] \fB-T\fR addrconf [\fB-i\fR \fIinterface-id\f
66 [\fB-p\fR {stateful|stateless}={yes|no},...] \fIaddrobj\fR
67 .fi

69 .LP
70 .nf
71 \fBipadm\fR down-addr [\fB-t\fR] \fIaddrobj\fR
72 .fi

74 .LP
75 .nf
76 \fBipadm\fR up-addr [\fB-t\fR] \fIaddrobj\fR
77 .fi

79 .LP
80 .nf
81 \fBipadm\fR disable-addr [\fB-t\fR] \fIaddrobj\fR
82 .fi

84 .LP
85 .nf
86 \fBipadm\fR enable-addr [\fB-t\fR] \fIaddrobj\fR
87 .fi

89 .LP
90 .nf
91 \fBipadm\fR refresh-addr [\fB-i\fR] \fIaddrobj\fR
92 .fi

94 .LP
95 .nf
96 \fBipadm\fR delete-addr [\fB-r\fR] \fIaddrobj\fR
97 .fi

99 .LP
100 .nf
101 \fBipadm\fR show-addr [[\fB-p\fR] \fB-o\fR \fIfield\fR[,...]] [\fIaddrobj\fR]
102 .fi

104 .LP
105 .nf
106 \fBipadm\fR set-addrprop [\fB-t\fR] \fB-p\fR \fIprop\fR=<\fIvalue\fR[,...]> \fIa
107 .fi

109 .LP
110 .nf
111 \fBipadm\fR reset-addrprop [\fB-t\fR] \fB-p\fR \fIprop\fR=<\fIvalue\fR[,...]> \f
112 .fi

114 .LP
115 .nf
116 \fBipadm\fR show-addrprop [[\fB-c\fR] \fB-o\fR \fIfield\fR[,...]] [\fB-p\fR \fIp
117 .fi

119 .LP
120 .nf
121 \fBipadm\fR set-prop [\fB-t\fR] \fB-p\fR \fIprop\fR[+|-]=<\fIvalue\fR[,...]> \fI
122 .fi

```

```

124 .LP
125 .nf
126 \fBipadm\fR reset-prop [\fB-t\fR] \fB-p\fR \fIprop\fR \fIprotocol\fR
127 .fi

129 .LP
130 .nf
131 \fBipadm\fR show-prop [[\fB-c\fR] \fB-o\fR \fIfield\fR[,...]] [\fB-p\fR \fIprop\
132 .fi

134 .SH DESCRIPTION
135 .sp
136 .LP

138 The \fBipadm\fR command is a stable replacement for the \fBifconfig\fR(1M) and
139 \fBndd\fR(1M) commands. It is used to create IP interfaces and to configure IP
140 addresses on those interfaces. It is also used to get, set or reset properties
141 on interfaces, addresses and protocols.
142 .LP
143 For subcommands that take an \fIaddrobj\fR, the \fIaddrobj\fR specifies a
144 unique address on the system. It is made up of two parts, delimited by a '/'.
145 The first part is the name of the interface and the second part is a string up
146 to 32 characters long. For example, "lo0/v4" is a loopback interface
147 addrobj name.
148 .LP
149 For subcommands that take a \fIprotocol\fR, this can be one of
150 the following values: ip, ipv4, ipv6, icmp, tcp, sctp or udp.

152 .SH SUBCOMMANDS
153 .sp
154 .LP
155 The following subcommands are supported:
156 .sp
157 .ne 2
158 .na
159 \fBifbcreate-if\fR [\fB-t\fR] \fIinterface\fR\fR
160 .ad
161 .sp .6
162 .RS 4n
163 The \fBcreate-if\fR subcommand is used to create an IP interface that will
164 handle both IPv4 and IPv6 packets. The interface will be enabled as part of
165 the creation process. The IPv4 interface will have the address 0.0.0.0.
166 The IPv6 interface will have the address ::.
167 .sp
168 The \fB-t\fR option (also \fB--temporary\fR) means
169 that the creation is temporary and will not be persistent across reboots.
170 .sp

172 .RE

174 .sp
175 .ne 2
176 .na
177 \fBifbdisable-if\fR [\fB-t\fR] \fIinterface\fR\fR
178 .ad
179 .sp .6
180 .RS 4n
181 The \fBdisable-if\fR subcommand is used to disable an IP interface.
182 .sp
183 The \fB-t\fR option (also \fB--temporary\fR) means
184 that the disable is temporary and will not be persistent across reboots.
185 .sp

187 .RE

```

```

189 .sp
190 .ne 2
191 .na
192 \fBifbenable-if\fR [\fB-t\fR] \fIinterface\fR\fR
193 .ad
194 .sp .6
195 .RS 4n
196 The \fBenable-if\fR subcommand is used to enable an IP interface.
197 .sp
198 The \fB-t\fR option (also \fB--temporary\fR) means
199 that the enable is temporary and will not be persistent across reboots.
200 .sp

202 .RE

204 .sp
205 .ne 2
206 .na
207 \fBifbdelete-if\fR \fIinterface\fR\fR
208 .ad
209 .sp .6
210 .RS 4n
211 The \fBdelete-if\fR subcommand is used to permanently delete an IP interface.
212 .sp

214 .RE

216 .sp
217 .ne 2
218 .na
219 \fBifbshow-if\fR [[\fB-p\fR] \fB-o\fR \fIfield\fR[,...]] [\fIinterface\fR]\fR
220 .ad
221 .sp .6
222 .RS 4n
223 The \fBshow-if\fR subcommand is used to show the current IP interface
224 configuration.
225 .sp
226 The \fB-p\fR option (also \fB--parsable\fR) prints
227 the output in a parsable format.
228 .sp
229 The \fB-o\fR option (also \fB--output\fR) is used
230 to select which fields will be shown. The field value can be one of the
231 following names:
232 .sp
233 .ne 2
234 .na
235 .RS 4n
236 \fBIBALL\fR
237 .ad
238 .RS 4n
239 Display all fields
240 .RE

242 .sp
243 .ne 2
244 .na
245 \fBIFNAME\fR
246 .ad
247 .RS 4n
248 The name of the interface
249 .RE

251 .sp
252 .ne 2
253 .na
254 \fBSTATE\fR

```

```

255 .ad
256 .RS 4n
257 The state can be one of the following values:
258 .sp
259 .ne 2
260 .na
261 .RS 4n
262 ok - resources for the interface have been allocated
263 .sp
264 offline - the interface is offline
265 .sp
266 failed - the interface's datalink is down
267 .sp
268 down - the interface is down
269 .sp
270 disabled - the interface is disabled
271 .RE
272 .RE

274 .sp
275 .ne 2
276 .na
277 \fBCURRENT\fR
278 .ad
279 .RS 4n
280 A set of single character flags indicating the following:
281 .sp
282 .ne 2
283 .na
284 .RS 4n
285 b - broadcast (mutually exclusive with 'p')
286 .br
287 m - multicast
288 .br
289 p - point-to-point (mutually exclusive with 'b')
290 .br
291 v - virtual interface
292 .br
293 I - IPMP
294 .br
295 s - IPMP standby
296 .br
297 i - IPMP inactive
298 .br
299 V - VRRP
300 .br
301 a - VRRP accept mode
302 .br
303 4 - IPv4
304 .br
305 6 - IPv6
306 .RE
307 .RE

309 .sp
310 .ne 2
311 .na
312 \fBPERSISTENT\fR
313 .ad
314 .RS 4n
315 A set of single character flags showing what configuration will be used the
316 next time the interface is enabled:
317 .sp
318 .ne 2
319 .na
320 .RS 4n

```

```

321 s - IPMP standby
322 .br
323 4 - IPv4
324 .br
325 6 - IPv6
326 .RE
327 .RE
328 .RE

330 .RE

332 .sp
333 .ne 2
334 .na
335 \fB\fBset-ifprop\fR [\fB-t\fR] \fB-p\fR \fIprop\fR=<\fIvalue\fR[,...]> \fB-m\fR
336 .ad
337 .sp .6
338 .RS 4n
339 The \fBset-ifprop\fR subcommand is used to set a property's value(s) on the IP
340 interface.
341 .sp
342 The \fB-t\fR option (also \fB--temporary\fR) means
343 that the setting is temporary and will not be persistent across reboots.
344 .sp
345 The \fB-p\fR option (also \fB--prop\fR) specifies the property name and
346 value(s). The property name can be one of the following:
347 .sp
348 .ne 2
349 .na

351 .RS 4n

353 \fBArp\fR
354 .ad
355 .RS 4n
356 Enables ("on") or disables ("off") ARP.
357 .RE

359 .sp
360 .ne 2
361 .na
362 \fBExchange_routes\fR
363 .ad
364 .RS 4n
365 Enables ("on") or disables ("off") the exchange of routing data.
366 .RE

368 .sp
369 .ne 2
370 .na
371 \fBForwarding\fR
372 .ad
373 .RS 4n
374 Enables ("on") or disables ("off") IP forwarding.
375 .RE

377 .sp
378 .ne 2
379 .na
380 \fBmetric\fR
381 .ad
382 .RS 4n
383 Set the routing metric to the numeric value. The value is treated as extra
384 hops to the destination.
385 .RE

```

```

387 .sp
388 .ne 2
389 .na
390 \fBmtu\fR
391 .ad
392 .RS 4n
393 Set the maximum transmission unit to the numeric value.
394 .RE

396 .sp
397 .ne 2
398 .na
399 \fBnud\fR
400 .ad
401 .RS 4n
402 Enables ("on") or disables ("off") neighbor unreachability detection.
403 .RE

405 .sp
406 .ne 2
407 .na
408 \fBusesrc\fR
409 .ad
410 .RS 4n
411 Indicates which interface to use for source address selection. A value
412 "none" may also be used.
413 .RE
414 .RE

416 .sp
417 The \fB-m\fR option (also \fB--module\fR) specifies which protocol
418 the setting applies to.
419 .sp

421 .RE
422 .RE

423 .sp
424 .ne 2
425 .na
426 \fB\breset-ifprop\fR [\fB-t\fR] \fB-p\fR \fIprop\fR \fB-m\fR \fIprotocol\fR \fI
427 .ad
428 .sp .6
429 .RS 4n
430 The \fBreset-ifprop\fR subcommand is used to reset an IP interface's property
431 value to the default.
432 .sp
433 The \fB-t\fR option (also \fB--temporary\fR) means
434 that the disable is temporary and will not be persistent across reboots.
435 .sp
436 The \fB-p\fR option (also \fB--prop\fR) specifies the property name.
437 See the \fBset-ifprop\fR subcommand for the list of property names.
438 .sp
439 The \fB-m\fR option (also \fB--module\fR) specifies which protocol
440 the setting applies to.
441 .sp

443 .RE

445 .sp
446 .ne 2
447 .na
448 \fB\bshow-ifprop\fR [[\fB-c\fR]\fB-o\fR \fIfield\fR[,...]] [\fB-p\fR \fIprop\fR
449 [\fIinterface\fR]\fR
450 .ad
451 .sp .6

```

```

452 .RS 4n
453 The \fBshow-ifprop\fR subcommand is used to display the property values
454 for one or all of the IP interfaces.
455 .sp
456 The \fB-c\fR option (also \fB--parsable\fR) prints
457 the output in a parsable format.
458 .sp
459 The \fB-o\fR option (also \fB--output\fR) is used
460 to select which fields will be shown. The field value can be one of the
461 following names:
462 .sp
463 .ne 2
464 .na
465 .RS 4n
466 \fBALL\fR
467 .ad
468 .RS 4n
469 Display all fields
470 .RE

472 .sp
473 .ne 2
474 .na
475 \fBIFNAME\fR
476 .ad
477 .RS 4n
478 The name of the interface
479 .RE

481 .sp
482 .ne 2
483 .na
484 \fBPROPERTY\fR
485 .ad
486 .RS 4n
487 The name of the property
488 .RE

490 .sp
491 .ne 2
492 .na
493 \fBPROTO\fR
494 .ad
495 .RS 4n
496 The name of the protocol
497 .RE

499 .sp
500 .ne 2
501 .na
502 \fBPERM\fR
503 .ad
504 .RS 4n
505 If the property is readable ("r") and/or writable ("w").
506 .RE

508 .sp
509 .ne 2
510 .na
511 \fBCURRENT\fR
512 .ad
513 .RS 4n
514 The value of the property
515 .RE

517 .sp

```

```

518 .ne 2
519 .na
520 \fBPERSISTENT\fR
521 .ad
522 .RS 4n
523 The persistent value of the property
524 .RE

526 .sp
527 .ne 2
528 .na
529 \fBDEFAULT\fR
530 .ad
531 .RS 4n
532 The default value of the property
533 .RE

535 .sp
536 .ne 2
537 .na
538 \fBPOSSIBLE\fR
539 .ad
540 .RS 4n
541 The possible values for the property
542 .RE
543 .RE

545 .sp
546 The \fB-p\fR option (also \fB--prop\fR) is used
547 to specify which properties to display. See the \fBset-ifprop\fR
548 subcommand for the list of property names.
549 .sp
550 The \fB-m\fR option (also \fB--module\fR) specifies which protocol
551 to display.
552 .sp

554 .RE

556 .sp
557 .ne 2
558 .na
559 \fB\fBcreate-addr\fR [\fB-t\fR] \fB-T\fR static [\fB-d\fR] \\  

560 \fB-a\fR {local|remote}=\fIaddr\fR[/\fIprefixlen\fR],... \fIaddrobj\fR\fR
561 .br
562 \fB\fBcreate-addr\fR [\fB-t\fR] \fB-T\fR dhcp [\fB-w\fR \fIseconds\fR | forever
563 .br
564 \fB\fBcreate-addr\fR [\fB-t\fR] \fB-T\fR addrconf [\fB-i\fR \fIinterface-id\fR]
565 [\fB-p\fR {stateful|stateless}={yes|no},...] \fIaddrobj\fR\fR
566 .ad
567 .sp .6
568 .RS 4n
569 The \fBcreate-addr\fR subcommand is used to set an address on an IP interface.
570 The address will be enabled but can disabled using the \fBdisable-addr\fR
571 subcommand. This subcommand has three different forms, depending on the
572 value of the \fB-T\fR option.
573 .sp
574 The \fB-t\fR option (also \fB--temporary\fR) means
575 that the address is temporary and will not be persistent across reboots.
576 .sp
577 The \fB-T\fR static option creates a static addrobj. This takes the following
578 options:
579 .RS 4n

581 The \fB-d\fR option (also \fB--down\fR) means the address is down.
582 .sp
583 The \fB-a\fR option (also \fB--address\fR) specifies the address.

```

```

584 The "local" or "remote" prefix can be used for a point-to-point interface.
585 In this case, both addresses must be given.
586 Otherwise, the equal sign ("=") should be omitted and the address should be
587 provided by itself and with no second address.
588 .sp

590 .RE

592 The \fB-T\fR dhcp option causes the address to be obtained via DHCP.
593 This takes the following options:
594 .RS 4n

596 The \fB-w\fR option (also \fB--wait\fR) gives the time, in seconds,
597 that the command should wait to obtain an address.
598 .sp

600 .RE

602 The \fB-T\fR addrconf option creates an auto-configured address.
603 This takes the following options:
604 .RS 4n

606 The \fB-i\fR option (also \fB--interface-id\fR) gives the interface ID to
607 be used.
608 .sp
609 The \fB-p\fR option (also \fB--prop\fR) indicates which method of
610 auto-configuration should be used.
611 .sp

613 .RE
614 .RE

616 .sp
617 .ne 2
618 .na
619 \fB\fBdown-addr\fR [\fB-t\fR] \fIaddrobj\fR\fR
620 .ad
621 .sp .6
622 .RS 4n
623 The \fBdown-addr\fR subcommand is used to down the address. This will
624 stop packets from being sent or received.
625 .sp
626 The \fB-t\fR option (also \fB--temporary\fR) means
627 that the down is temporary and will not be persistent across reboots.
628 .sp

630 .RE

632 .sp
633 .ne 2
634 .na
635 \fB\fBup-addr\fR [\fB-t\fR] \fIaddrobj\fR\fR
636 .ad
637 .sp .6
638 .RS 4n
639 The \fBup-addr\fR subcommand is used to up the address. This will
640 enable packets to be sent and received.
641 .sp
642 The \fB-t\fR option (also \fB--temporary\fR) means
643 that the up is temporary and will not be persistent across reboots.
644 .sp

646 .RE

648 .sp
649 .ne 2

```

```

650 .na
651 \fB\fBdisable-addr\fR [\fB-t\fR] \fIiaddrobj\fR\fR
652 .ad
653 .sp .6
654 .RS 4n
655 The \fBdisable-addr\fR subcommand is used to disable the address.
656 .sp
657 The \fB-t\fR option (also \fB--temporary\fR) means
658 that the disable is temporary and will not be persistent across reboots.
659 .sp

661 .RE

663 .sp
664 .ne 2
665 .na
666 \fB\fBenable-addr\fR [\fB-t\fR] \fIiaddrobj\fR\fR
667 .ad
668 .sp .6
669 .RS 4n
670 The \fBenable-addr\fR subcommand is used to enable the address.
671 .sp
672 The \fB-t\fR option (also \fB--temporary\fR) means
673 that the enable is temporary and will not be persistent across reboots.
674 .sp

676 .RE

678 .sp
679 .ne 2
680 .na
681 \fB\fBrefresh-addr\fR [\fB-i\fR] \fIiaddrobj\fR\fR
682 .ad
683 .sp .6
684 .RS 4n
685 The \fBrefresh-addr\fR subcommand is used to extend the lease for DHCP
686 addresses. It also restarts duplicate address detection for Static addresses.
687 .sp
688 The \fB-i\fR option (also \fB--inform\fR) means
689 that the network configuration will be obtained from DHCP without taking
690 a lease on the address.
691 .sp

693 .RE

695 .sp
696 .ne 2
697 .na
698 \fB\fBdelete-addr\fR [\fB-r\fR] \fIiaddrobj\fR\fR
699 .ad
700 .sp .6
701 .RS 4n
702 The \fBdelete-addr\fR subcommand deletes the given address.
703 .sp
704 The \fB-r\fR option (also \fB--release\fR) is used for DHCP-assigned
705 addresses to indicate that the address should be released.
706 .sp

708 .RE

710 .sp
711 .ne 2
712 .na
713 \fB\fBshow-addr\fR [[\fB-p\fR] \fB-o\fR \fIifield\fR[,...]] [\fIiaddrobj\fR]\fR
714 .ad
715 .sp .6

```

```

716 .RS 4n
717 The \fBshow-addr\fR subcommand is used to show the current address properties.
718 .sp
719 The \fB-p\fR option (also \fB--parsable\fR) prints
720 the output in a parsable format.
721 .sp
722 The \fB-o\fR option (also \fB--output\fR) is used
723 to select which fields will be shown. The field value can be one of the
724 following names:
725 .sp
726 .ne 2
727 .na
728 .RS 4n
729 \fBALL\fR
730 .ad
731 .RS 4n
732 Display all fields
733 .RE

735 .sp
736 .ne 2
737 .na
738 \fBADDROBJ\fR
739 .ad
740 .RS 4n
741 The name of the address
742 .RE

744 .sp
745 .ne 2
746 .na
747 \fBTYPE\fR
748 .ad
749 .RS 4n
750 The type of the address. It can be "static", "dhcp" or "addrconf".
751 .RE

753 .sp
754 .ne 2
755 .na
756 \fBSTATE\fR
757 .ad
758 .RS 4n
759 The state of the address. It can be one of the following values:
760 .sp
761 .ne 2
762 .na
763 .RS 4n
764 disabled s see the \fBdisable-addr\fR subcommand
765 .sp
766 down - see the \fBdown-addr\fR subcommand
767 .sp
768 duplicate - the address is a duplicate
769 .sp
770 inaccessible - the interface for this address has failed
771 .sp
772 ok - the address is up
773 .sp
774 tentative - duplicate address detection in progress
775 .RE
776 .RE

778 .sp
779 .ne 2
780 .na
781 \fBCURRENT\fR

```

```

782 .ad
783 .RS 4n
784 A set of single character flags indicating the following:
785 .sp
786 .ne 2
787 .na
788 .RS 4n
789 U - up
790 .br
791 u - unnumbered (matches another local address)
792 .br
793 p - private, not advertised to routing
794 .br
795 t - temporary IPv6 address
796 .br
797 d - deprecated (not used for outgoing packets)
798 .RE
799 .RE

801 .sp
802 .ne 2
803 .na
804 \fBPERSISTENT\fR
805 .ad
806 .RS 4n
807 A set of single character flags showing the configuration which will be used
808 when the address is enabled.
809 .sp
810 .ne 2
811 .na
812 .RS 4n
813 U - up
814 .br
815 p - private, not advertised to routing
816 .br
817 d - deprecated (not used for outgoing packets)
818 .RE
819 .RE

821 .sp
822 .ne 2
823 .na
824 \fBADDR\fR
825 .ad
826 .RS 4n
827 The address
828 .RE
829 .RE

831 .RE

833 .sp
834 .ne 2
835 .na
836 \fB\fBset-addrprop\fR [\fB-t\fR] \fB-p\fR \fIprop\fR=<\fIvalue\fR[,...]> \fIaddr
837 .ad
838 .sp .6
839 .RS 4n
840 The \fBset-addrprop\fR subcommand is used to set a property's value(s) on the
841 addrobj.
842 .sp
843 The \fB-t\fR option (also \fB--temporary\fR) means
844 that the setting is temporary and will not be persistent across reboots.
845 .sp
846 The \fB-p\fR option (also \fB--prop\fR) specifies the property name and
847 value(s). The property name can be one of the following:

```

```

848 .sp
849 .ne 2
850 .na

852 .RS 4n

854 \fBbroadcast\fR
855 .ad
856 .RS 4n
857 The broadcast address (read-only)
858 .RE

860 .sp
861 .ne 2
862 .na
863 \fBdeprecated\fR
864 .ad
865 .RS 4n
866 The address should not be used to send packets but can still receive packets.
867 Can be "on" or "off".
868 .RE

870 .sp
871 .ne 2
872 .na
873 \fBprefixlen\fR
874 .ad
875 .RS 4n
876 The number of bits in the IPv4 netmask or IPv6 prefix.
877 .RE

879 .sp
880 .ne 2
881 .na
882 \fBprivate\fR
883 .ad
884 .RS 4n
885 The address is not advertised to routing.
886 Can be "on" or "off".
887 .RE

889 .sp
890 .ne 2
891 .na
892 \fBtransmit\fR
893 .ad
894 .RS 4n
895 Packets can be transmitted.
896 Can be "on" or "off".
897 .RE

899 .sp
900 .ne 2
901 .na
902 \fBzone\fR
903 .ad
904 .RS 4n
905 The zone the addrobj is in.
906 .RE

908 .RE
909 .RE

911 .sp
912 .ne 2
913 .na

```



```

914 \fb\fbreset-addrprop\fr [[\fb-t\fr] \fb-p\fr \fiIprop\fr \fiIaddrobj\fr\fr
915 .ad
916 .sp .6
917 .RS 4n
918 The \fbreset-addrprop\fr subcommand is used to reset an addrobj's property
919 value to the default.
920 .sp
921 The \fb-t\fr option (also \fb--temporary\fr) means
922 that the disable is temporary and will not be persistent across reboots.
923 .sp
924 The \fb-p\fr option (also \fb--prop\fr) specifies the property name.
925 See the \fbset-addrprop\fr subcommand for the list of property names.
926 .sp

928 .RE

930 .sp
931 .ne 2
932 .na
933 \fb\fbshow-addrprop\fr [[\fb-c\fr]\fb-o\fr \fiIfield\fr[,...]] [\fb-p\fr \fiIprop\
934 .ad
935 .sp .6
936 .RS 4n
937 The \fbshow-addrprop\fr subcommand is used to display the property values
938 for one or all of the addrobjs.
939 .sp
940 The \fb-c\fr option (also \fb--parsable\fr) prints
941 the output in a parsable format.
942 .sp
943 The \fb-o\fr option (also \fb--output\fr) is used
944 to select which fields will be shown. The field value can be one of the
945 following names:
946 .sp
947 .ne 2
948 .na
949 .RS 4n
950 \fbALL\fr
951 .ad
952 .RS 4n
953 Display all fields
954 .RE

956 .sp
957 .ne 2
958 .na
959 \fbADDROBJ\fr
960 .ad
961 .RS 4n
962 The name of the addrobj
963 .RE

965 .sp
966 .ne 2
967 .na
968 \fbPROPERTY\fr
969 .ad
970 .RS 4n
971 The name of the property
972 .RE

974 .sp
975 .ne 2
976 .na
977 \fbPERM\fr
978 .ad
979 .RS 4n

```

```

980 If the property is readable ("r") and/or writable ("w").
981 .RE

983 .sp
984 .ne 2
985 .na
986 \fbCURRENT\fr
987 .ad
988 .RS 4n
989 The value of the property
990 .RE

992 .sp
993 .ne 2
994 .na
995 \fbPERSISTENT\fr
996 .ad
997 .RS 4n
998 The persistent value of the property
999 .RE

1001 .sp
1002 .ne 2
1003 .na
1004 \fbDEFAULT\fr
1005 .ad
1006 .RS 4n
1007 The default value of the property
1008 .RE

1010 .sp
1011 .ne 2
1012 .na
1013 \fbPOSSIBLE\fr
1014 .ad
1015 .RS 4n
1016 The possible values for the property
1017 .RE
1018 .RE

1020 .sp
1021 The \fb-p\fr option (also \fb--prop\fr) is used
1022 to specify which properties to display. See the \fbset-addrprop\fr
1023 subcommand for the list of property names.
1024 .sp

1026 .RE

1028 .sp
1029 .ne 2
1030 .na
1031 \fb\fbset-prop\fr [[\fb-t\fr] \fb-p\fr \fiIprop\fr[+|-]=<\fiIvalue\fr[,...]> \fiIpro
1032 .ad
1033 .sp .6
1034 .RS 4n
1035 The \fbset-prop\fr subcommand is used to set a property's value(s) on the
1036 protocol.
1037 .sp
1038 The \fb-t\fr option (also \fb--temporary\fr) means
1039 that the setting is temporary and will not be persistent across reboots.
1040 .sp
1041 The \fb-p\fr option (also \fb--prop\fr) specifies the property name and
1042 value(s). The optional [+|-] syntax can be used to add/remove values from the
1043 current list of values on the property.
1044 The property name can be one of the following:
1045 .sp

```

```

1046 .ne 2
1047 .na

1049 .RS 4n

1051 \fBecn\fR
1052 .ad
1053 .RS 4n
1054 Explicit congestion control (TCP-only)
1055 Can be "never", "passive" or "active".
1056 .RE

1058 \fBextra_priv_ports\fR
1059 .ad
1060 .RS 4n
1061 Additional privileged ports (SCTP, TCP or UDP)
1062 .RE

1064 \fBforwarding\fR
1065 .ad
1066 .RS 4n
1067 Packet forwarding is enabled.
1068 Can be "on" or "off".
1069 .RE

1071 \fBhoplimit\fR
1072 .ad
1073 .RS 4n
1074 The IPv6 hoplimit.
1075 .RE

1077 \fBlargest_anon_port\fR
1078 .ad
1079 .RS 4n
1080 Largest ephemeral port (SCTP, TCP or UDP)
1081 .RE

1083 \fBmax_buf\fR
1084 .ad
1085 .RS 4n
1086 Maximum receive or send buffer size (ICMP, SCTP, TCP, or UDP). This also
1087 sets the upper limit for the \fBrecv_buf\fR and \fBsend_buf\fR properties.
1088 .RE

1090 \fBrecv_buf\fR
1091 .ad
1092 .RS 4n
1093 Default receive buffer size (ICMP, SCTP, TCP, or UDP). The maximum value for
1094 this property is controlled by the \fBmax_buf\fR property.
1095 .RE

1097 \fBsack\fR
1098 .ad
1099 .RS 4n
1100 Selective acknowledgement (TCP).
1101 Can be "active", "passive" or "never".
1102 .RE

1104 \fBsend_buf\fR
1105 .ad
1106 .RS 4n
1107 Default send buffer size (ICMP, SCTP, TCP, or UDP). The maximum value for
1108 this property is controlled by the \fBmax_buf\fR property.
1109 .RE

1111 \fBsmallest_anon_port\fR

```

```

1112 .ad
1113 .RS 4n
1114 Smallest ephemeral port (SCTP, TCP or UDP)
1115 .RE

1117 \fBsmallest_nonpriv_port\fR
1118 .ad
1119 .RS 4n
1120 Smallest non-privileged port (SCTP, TCP or UDP)
1121 .RE

1123 \fBttl\fR
1124 .ad
1125 .RS 4n
1126 The IPv4 time-to-live.
1127 .RE

1129 .RE
1130 .RE

1132 .sp
1133 .ne 2
1134 .na
1135 \fB\fBreset-prop\fR [\fB-t\fR] \fB-p\fR \fIprop\fR \fIprotocol\fR\fR
1136 .ad
1137 .sp .6
1138 .RS 4n
1139 The \fBreset-prop\fR subcommand is used to reset a protocol's property
1140 value to the default.
1141 .sp
1142 The \fB-t\fR option (also \fB--temporary\fR) means
1143 that the disable is temporary and will not be persistent across reboots.
1144 .sp
1145 The \fB-p\fR option (also \fB--prop\fR) specifies the property name.
1146 See the \fBset-prop\fR subcommand for the list of property names.
1147 .sp

1149 .RE

1151 .sp
1152 .ne 2
1153 .na
1154 \fB\fBshow-prop\fR [[\fB-c\fR]\fB-o\fR \fIfield\fR[,...]] [\fB-p\fR \fIprop\fR,.
1155 .ad
1156 .sp .6
1157 .RS 4n
1158 The \fBshow-prop\fR subcommand is used to display the property values
1159 for one or all of the protocols.
1160 .sp
1161 The \fB-c\fR option (also \fB--parsable\fR) prints
1162 the output in a parsable format.
1163 .sp
1164 The \fB-o\fR option (also \fB--output\fR) is used
1165 to select which fields will be shown. The field value can be one of the
1166 following names:
1167 .sp
1168 .ne 2
1169 .na
1170 .RS 4n
1171 \fBALL\fR
1172 .ad
1173 .RS 4n
1174 Display all fields
1175 .RE

1177 .sp

```

```

1178 .ne 2
1179 .na
1180 \fBPROTO\fR
1181 .ad
1182 .RS 4n
1183 The name of the protocol
1184 .RE

1186 .sp
1187 .ne 2
1188 .na
1189 \fBPROPERTY\fR
1190 .ad
1191 .RS 4n
1192 The name of the property
1193 .RE

1195 .sp
1196 .ne 2
1197 .na
1198 \fBPERM\fR
1199 .ad
1200 .RS 4n
1201 If the property is readable ("r") and/or writable ("w").
1202 .RE

1204 .sp
1205 .ne 2
1206 .na
1207 \fBCURRENT\fR
1208 .ad
1209 .RS 4n
1210 The value of the property
1211 .RE

1213 .sp
1214 .ne 2
1215 .na
1216 \fBPERSISTENT\fR
1217 .ad
1218 .RS 4n
1219 The persistent value of the property
1220 .RE

1222 .sp
1223 .ne 2
1224 .na
1225 \fBDEFAULT\fR
1226 .ad
1227 .RS 4n
1228 The default value of the property
1229 .RE

1231 .sp
1232 .ne 2
1233 .na
1234 \fBPOSSIBLE\fR
1235 .ad
1236 .RS 4n
1237 The possible values for the property
1238 .RE
1239 .RE

1241 .sp
1242 The \fB-p\fR option (also \fB--prop\fR) is used
1243 to specify which properties to display. See the \fBset-prop\fR

```

```

1244 subcommand for the list of property names.
1245 .sp

1247 .RE

1249 .SH SEE ALSO
1250 .sp
1251 .LP
1252 \fBifconfig\fR(1M), \fBdladm\fR(1M), \fBndd\fR(1M), \fBzonecfg\fR(1M),
1253 \fBbarp\fR(1M), \fBcfgadm\fR(1M), \fBifmpadm\fR(1M), \fBnsswitch.conf\fR(4),
1254 and \fBdhcp\fR(5).

```

18373 Sat Jul 19 19:24:35 2014

new/usr/src/man/man1m/lofiadm.1m

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 \" te
2 .\" Copyright 2013 Nexenta Systems, Inc. All rights reserved.
3 .\" Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved
4 .\" The contents of this file are subject to the terms of the Common Development
5 .\" See the License for the specific language governing permissions and limitat
6 .\" the fields enclosed by brackets \"[]\" replaced with your own identifying info
7 .TH LOFIADM 1M \"Aug 28, 2013\"
8 .SH NAME
9 lofiadm \- administer files available as block devices through lofi
10 .SH SYNOPSIS
11 .LP
12 .nf
13 \fBlofiadm\fR [\fB-r\fR] \fB-a\fR \fIfile\fR [\fIdevice\fR]
14 .fi

16 .LP
17 .nf
18 \fBlofiadm\fR [\fB-r\fR] \fB-c\fR \fIcrypto_algorithm\fR \fB-a\fR \fIfile\fR [\f
19 .fi

21 .LP
22 .nf
23 \fBlofiadm\fR [\fB-r\fR] \fB-c\fR \fIcrypto_algorithm\fR \fB-k\fR \fIraw_key_fil
24 .fi

26 .LP
27 .nf
28 \fBlofiadm\fR [\fB-r\fR] \fB-c\fR \fIcrypto_algorithm\fR \fB-T\fR \fItoken_key\f
29 .fi

31 .LP
32 .nf
33 \fBlofiadm\fR [\fB-r\fR] \fB-c\fR \fIcrypto_algorithm\fR \fB-T\fR \fItoken_key\f
34 \fB-k\fR \fIwrapped_key_file\fR \fB-a\fR \fIfile\fR [\fIdevice\fR]
35 .fi

37 .LP
38 .nf
39 \fBlofiadm\fR [\fB-r\fR] \fB-c\fR \fIcrypto_algorithm\fR \fB-e\fR \fB-a\fR \fIfi
40 .fi

42 .LP
43 .nf
44 \fBlofiadm\fR \fB-C\fR \fIalgorithm\fR [\fB-s\fR \fIsegment_size\fR] \fIfile\fR
45 .fi

47 .LP
48 .nf
49 \fBlofiadm\fR \fB-d\fR \fIfile\fR | \fIdevice\fR
50 .fi

52 .LP
53 .nf
54 \fBlofiadm\fR \fB-U\fR \fIfile\fR
55 .fi

```

```

57 .LP
58 .nf
59 \fBlofiadm\fR [ \fIfile\fR | \fIdevice\fR]
60 .fi

62 .SH DESCRIPTION
63 .sp
64 .LP
65 \fBlofiadm\fR administers \fBlofi\fR, the loopback file driver. \fBlofi\fR
66 allows a file to be associated with a block device. That file can then be
67 accessed through the block device. This is useful when the file contains an
68 image of some filesystem (such as a floppy or \fBCD-ROM\fR image), because the
69 block device can then be used with the normal system utilities for mounting,
70 checking or repairing filesystems. See \fBfsck\fR(1M) and \fBmount\fR(1M).
71 .sp
72 .LP
73 Use \fBlofiadm\fR to add a file as a loopback device, remove such an
74 association, or print information about the current associations.
75 .sp
76 .LP
77 Encryption and compression options are mutually exclusive on the command line.
78 Further, an encrypted file cannot be compressed later, nor can a compressed
79 file be encrypted later.

81 In the global zone, \fBlofiadm\fR can be used on both the global
82 zone devices and all devices owned by other non-global zones on the system.
83 .sp
84 .LP
84 .SH OPTIONS
85 .sp
86 .LP
87 The following options are supported:
88 .sp
89 .ne 2
90 .na
91 \fBlofiadm\fR \fB-a\fR \fIfile\fR [\fIdevice\fR]\fR
92 .ad
93 .sp .6
94 .RS 4n
95 Add \fIfile\fR as a block device.
96 .sp
97 If \fIdevice\fR is not specified, an available device is picked.
98 .sp
99 If \fIdevice\fR is specified, \fBlofiadm\fR attempts to assign it to
100 \fIfile\fR. \fIdevice\fR must be available or \fBlofiadm\fR will fail. The
101 ability to specify a device is provided for use in scripts that wish to
102 reestablish a particular set of associations.
103 .RE

105 .sp
106 .ne 2
107 .na
108 \fBlofiadm\fR \fB-C\fR {\fIgzip\fR | \fIgzip-N\fR | \fIlzma\fR}\fR
109 .ad
110 .sp .6
111 .RS 4n
112 Compress the file with the specified compression algorithm.
113 .sp
114 The \fBgzip\fR compression algorithm uses the same compression as the
115 open-source \fBgzip\fR command. You can specify the \fBgzip\fR level by using
116 the value \fBgzip-\fR\fIN\fR where \fIN\fR is 6 (fast) or 9 (best compression
117 ratio). Currently, \fBgzip\fR, without a number, is equivalent to \fBgzip-6\fR
118 (which is also the default for the \fBgzip\fR command).
119 .sp
120 \fIlzma\fR stands for the LZMA (Lempel-Ziv-Markov) compression algorithm.
121 .sp

```

```

122 Note that you cannot write to a compressed file, nor can you mount a compressed
123 file read/write.
124 .RE

126 .sp
127 .ne 2
128 .na
129 \fB\fB-d\fR \fIfile\fR | \fIdevice\fR\fR
130 .ad
131 .sp .6
132 .RS 4n
133 Remove an association by \fIfile\fR or \fIdevice\fR name, if the associated
134 block device is not busy, and deallocates the block device.
135 .RE

137 .sp
138 .ne 2
139 .na
140 \fB\fB-r\fR
141 .ad
142 .sp .6
143 .RS 4n
144 If the \fB-r\fR option is specified before the \fB-a\fR option, the
145 \fIdevice\fR will be opened read-only.
146 .RE

148 .sp
149 .ne 2
150 .na
151 \fB\fB-s\fR \fIsegment_size\fR\fR
152 .ad
153 .sp .6
154 .RS 4n
155 The segment size to use to divide the file being compressed. \fIsegment_size\fR
156 can be an integer multiple of 512.
157 .RE

159 .sp
160 .ne 2
161 .na
162 \fB\fB-U\fR \fIfile\fR\fR
163 .ad
164 .sp .6
165 .RS 4n
166 Uncompress a compressed file.
167 .RE

169 .sp
170 .LP
171 The following options are used when the file is encrypted:
172 .sp
173 .ne 2
174 .na
175 \fB\fB-c\fR \fIcrypto_algorithm\fR\fR
176 .ad
177 .sp .6
178 .RS 4n
179 Select the encryption algorithm. The algorithm must be specified when
180 encryption is enabled because the algorithm is not stored in the disk image.
181 .sp
182 If none of \fB-e\fR, \fB-k\fR, or \fB-T\fR is specified, \fB-lofiadm\fR prompts
183 for a passphrase, with a minimum length of eight characters, to be entered .
184 The passphrase is used to derive a symmetric encryption key using PKCS#5 PBKD2.
185 .RE

187 .sp

```

```

188 .ne 2
189 .na
190 \fB\fB-k\fR \fIraw_key_file\fR | \fIwrapped_key_file\fR\fR
191 .ad
192 .sp .6
193 .RS 4n
194 Path to raw or wrapped symmetric encryption key. If a PKCS#11 object is also
195 given with the \fB-T\fR option, then the key is wrapped by that object. If
196 \fB-T\fR is not specified, the key is used raw.
197 .RE

199 .sp
200 .ne 2
201 .na
202 \fB\fB-T\fR \fItoken_key\fR\fR
203 .ad
204 .sp .6
205 .RS 4n
206 The key in a PKCS#11 token to use for the encryption or for unwrapping the key
207 file.
208 .sp
209 If \fB-k\fR is also specified, \fB-T\fR identifies the unwrapping key, which
210 must be an RSA private key.
211 .RE

213 .sp
214 .ne 2
215 .na
216 \fB\fB-e\fR\fR
217 .ad
218 .sp .6
219 .RS 4n
220 Generate an ephemeral symmetric encryption key.
221 .RE

223 .SH OPERANDS
224 .sp
225 .LP
226 The following operands are supported:
227 .sp
228 .ne 2
229 .na
230 \fB\fIcrypto_algorithm\fR\fR
231 .ad
232 .sp .6
233 .RS 4n
234 One of: \fB-baes-128-cbc\fR, \fB-baes-192-cbc\fR, \fB-baes-256-cbc\fR,
235 \fB-bdes3-cbc\fR, \fB-blowfish-cbc\fR.
236 .RE

238 .sp
239 .ne 2
240 .na
241 \fB\fIdevice\fR\fR
242 .ad
243 .sp .6
244 .RS 4n
245 Display the file name associated with the block device \fIdevice\fR.
246 .sp
247 Without arguments, print a list of the current associations. Filenames must be
248 valid absolute pathnames.
249 .sp
250 When a file is added, it is opened for reading or writing by root. Any
251 restrictions apply (such as restricted root access over \fB-BNFS\fR). The file is
252 held open until the association is removed. It is not actually accessed until
253 the block device is used, so it will never be written to if the block device is

```

```

254 only opened read-only.

256 Note that the filename may appear as "?" if it is not possible to resolve the
257 path in the current context (for example, if it's an NFS path in a non-global
258 zone).
259 .RE

261 .sp
262 .ne 2
263 .na
264 \fB\fifile\fR\fR
265 .ad
266 .sp .6
267 .RS 4n
268 Display the block device associated with \fifile\fR.
269 .RE

271 .sp
272 .ne 2
273 .na
274 \fB\fIraw_key_file\fR\fR
275 .ad
276 .sp .6
277 .RS 4n
278 Path to a file of the appropriate length, in bits, to use as a raw symmetric
279 encryption key.
280 .RE

282 .sp
283 .ne 2
284 .na
285 \fB\fItoken_key\fR\fR
286 .ad
287 .sp .6
288 .RS 4n
289 PKCS#11 token object in the format:
290 .sp
291 .in +2
292 .nf
293 \fItoken_name\fR:\fImanufacturer_id\fR:\fIserial_number\fR:\fIkey_label\fR
294 .fi
295 .in -2
296 .sp

298 All but the key label are optional and can be empty. For example, to specify a
299 token object with only its key label \fBMylofiKey\fR, use:
300 .sp
301 .in +2
302 .nf
303 -T :::MylofiKey
304 .fi
305 .in -2
306 .sp

308 .RE

310 .sp
311 .ne 2
312 .na
313 \fB\fIwrapped_key_file\fR\fR
314 .ad
315 .sp .6
316 .RS 4n
317 Path to file containing a symmetric encryption key wrapped by the RSA private
318 key specified by \fB-T\fR.
319 .RE

```

```

321 .SH EXAMPLES
322 .LP
323 \fBExample 1 \fRMounting an Existing CD-ROM Image
324 .sp
325 .LP
326 You should ensure that Solaris understands the image before creating the
327 \fBCD\fR. \fBlofi\fR allows you to mount the image and see if it works.

329 .sp
330 .LP
331 This example mounts an existing \fBCD-ROM\fR image (\fBsparc.iso\fR), of the
332 \fBRed Hat 6.0 CD\fR which was downloaded from the Internet. It was created
333 with the \fBmkisofs\fR utility from the Internet.

335 .sp
336 .LP
337 Use \fBlofiadm\fR to attach a block device to it:

339 .sp
340 .in +2
341 .nf
342 # \fBlofiadm -a /home/mike_s/RH6.0/sparc.iso\fR
343 /dev/lofi/1
344 .fi
345 .in -2
346 .sp

348 .sp
349 .LP
350 \fBlofiadm\fR picks the device and prints the device name to the standard
351 output. You can run \fBlofiadm\fR again by issuing the following command:

353 .sp
354 .in +2
355 .nf
356 # \fBlofiadm\fR
357 Block Device      File                               Options
358 /dev/lofi/1      /home/mike_s/RH6.0/sparc.iso     -
359 .fi
360 .in -2
361 .sp

363 .sp
364 .LP
365 Or, you can give it one name and ask for the other, by issuing the following
366 command:

368 .sp
369 .in +2
370 .nf
371 # \fBlofiadm /dev/lofi/1\fR
372 /home/mike_s/RH6.0/sparc.iso
373 .fi
374 .in -2
375 .sp

377 .sp
378 .LP
379 Use the \fBmount\fR command to mount the image:

381 .sp
382 .in +2
383 .nf
384 # \fBmount -F hsfs -o ro /dev/lofi/1 /mnt\fR
385 .fi

```

```

386 .in -2
387 .sp

389 .sp
390 .LP
391 Check to ensure that Solaris understands the image:

393 .sp
394 .in +2
395 .nf
396 # \fBdf -k /mnt\fR
397 Filesystem          kbytes    used    avail capacity  Mounted on
398 /dev/lofi/1         512418   512418         0    100%    /mnt
399 # \fBls /mnt\fR
400 \&./                RedHat/    doc/      ls-lR      rr_moved/
401 \&./                TRANS.TBL  dosutils/ ls-lR.gz   sbin@
402 \&.buildlog        bin@       etc@      misc/      tmp/
403 COPYING            boot/      images/   mnt/      usr@
404 README             boot.cat*  kernels/  modules/
405 RPM-PGP-KEY        dev@      lib@      proc/
406 .fi
407 .in -2
408 .sp

410 .sp
411 .LP
412 Solaris can mount the CD-ROM image, and understand the filenames. The image was
413 created properly, and you can now create the \fBCD-ROM\fR with confidence.

415 .sp
416 .LP
417 As a final step, unmount and detach the images:

419 .sp
420 .in +2
421 .nf
422 # \fBumount /mnt\fR
423 # \fBlofiadm -d /dev/lofi/1\fR
424 # \fBlofiadm\fR
425 Block Device          File          Options
426 .fi
427 .in -2
428 .sp

430 .LP
431 \fBExample 2 \fRMounting a Floppy Image
432 .sp
433 .LP
434 This is similar to the first example.

436 .sp
437 .LP
438 Using \fBlofi\fR to help you mount files that contain floppy images is helpful
439 if a floppy disk contains a file that you need, but the machine which you are
440 on does not have a floppy drive. It is also helpful if you do not want to take
441 the time to use the \fBdd\fR command to copy the image to a floppy.

443 .sp
444 .LP
445 This is an example of getting to \fBMBD\fR floppy for Solaris on an x86
446 platform:

448 .sp
449 .in +2
450 .nf
451 # \fBlofiadm -a /export/s28/MDB_s28x_wos/latest/boot.3\fR

```

```

452 /dev/lofi/1
453 # \fBmount -F pcfs /dev/lofi/1 /mnt\fR
454 # \fBls /mnt\fR
455 \&./                COMMENT.BAT*  RC.D/      SOLARIS.MAP*
456 \&./                IDENT*        REPLACE.BAT* X/
457 APPEND.BAT*        MAKEDIR.BAT*  SOLARIS/
458 # \fBumount /mnt\fR
459 # \fBlofiadm -d /export/s28/MDB_s28x_wos/latest/boot.3\fR
460 .fi
461 .in -2
462 .sp

464 .LP
465 \fBExample 3 \fRMaking a \fBUFS\fR Filesystem on a File
466 .sp
467 .LP
468 Making a \fBUFS\fR filesystem on a file can be useful, particularly if a test
469 suite requires a scratch filesystem. It can be painful (or annoying) to have to
470 repartition a disk just for the test suite, but you do not have to. You can
471 \fBnewfs\fR a file with \fBlofi\fR

473 .sp
474 .LP
475 Create the file:

477 .sp
478 .in +2
479 .nf
480 # \fBmkfile 35m /export/home/test\fR
481 .fi
482 .in -2
483 .sp

485 .sp
486 .LP
487 Attach it to a block device. You also get the character device that \fBnewfs\fR
488 requires, so \fBnewfs\fR that:

490 .sp
491 .in +2
492 .nf
493 # \fBlofiadm -a /export/home/test\fR
494 /dev/lofi/1
495 # \fBnewfs /dev/rlofi/1\fR
496 newfs: construct a new file system /dev/rlofi/1: (y/n)? \fBy\fR
497 /dev/rlofi/1: 71638 sectors in 119 cylinders of 1 tracks, 602 sectors
498 35.0MB in 8 cyl groups (16 c/g, 4.70MB/g, 2240 i/g)
499 super-block backups (for fsck -F ufs -o b=#) at:
500 32, 9664, 19296, 28928, 38560, 48192, 57824, 67456,
501 .fi
502 .in -2
503 .sp

505 .sp
506 .LP
507 Note that \fBuffs\fR might not be able to use the entire file. Mount and use the
508 filesystem:

510 .sp
511 .in +2
512 .nf
513 # \fBmount /dev/lofi/1 /mnt\fR
514 # \fBdf -k /mnt\fR
515 Filesystem          kbytes    used    avail capacity  Mounted on
516 /dev/lofi/1         33455     9     30101     1%    /mnt
517 # \fBls /mnt\fR

```

```

518 \&./          ../          lost+found/
519 # \fBumount /mnt\fR
520 # \fBlofiadm -d /dev/lofi/1\fR
521 .fi
522 .in -2
523 .sp

525 .LP
526 \fBExample 4 \fRCreating a PC (FAT) File System on a Unix File
527 .sp
528 .LP
529 The following series of commands creates a \fBFAT\fR file system on a Unix
530 file. The file is associated with a block device created by \fBlofiadm\fR.

532 .sp
533 .in +2
534 .nf
535 # \fBmkfile 10M /export/test/testfs\fR
536 # \fBlofiadm -a /export/test/testfs\fR
537 /dev/lofi/1
538 \fBNote use of\fR rlofi\fB, not\fR lofi\fB, in following command.\fR
539 # \fBmkfs -F pcfs -o nofdisk,size=20480 /dev/rlofi/1\fR
540 \fBConstruct a new FAT file system on /dev/rlofi/1: (y/n)?\fR y
541 # \fBmount -F pcfs /dev/lofi/1 /mnt\fR
542 # \fBcd /mnt\fR
543 # \fBdf -k .\fR
544 Filesystem          kbytes    used   avail capacity  Mounted on
545 /dev/lofi/1         10142      0   10142    0%   /mnt
546 .fi
547 .in -2
548 .sp

550 .LP
551 \fBExample 5 \fRCompressing an Existing CD-ROM Image
552 .sp
553 .LP
554 The following example illustrates compressing an existing CD-ROM image
555 (\fBsolaris.iso\fR), verifying that the image is compressed, and then
556 uncompressing it.

558 .sp
559 .in +2
560 .nf
561 # \fBlofiadm -C gzip /export/home/solaris.iso\fR
562 .fi
563 .in -2
564 .sp

566 .sp
567 .LP
568 Use \fBlofiadm\fR to attach a block device to it:

570 .sp
571 .in +2
572 .nf
573 # \fBlofiadm -a /export/home/solaris.iso\fR
574 /dev/lofi/1
575 .fi
576 .in -2
577 .sp

579 .sp
580 .LP
581 Check if the mapped image is compressed:

583 .sp

```

```

584 .in +2
585 .nf
586 # \fBlofiadm \fR
587 Block Device      File      Options
588 /dev/lofi/1       /export/home/solaris.iso  Compressed(gzip)
589 /dev/lofi/2       /export/home/regular.iso   -
590 .fi
591 .in -2
592 .sp

594 .sp
595 .LP
596 Unmap the compressed image and uncompress it:

598 .sp
599 .in +2
600 .nf
601 # \fBlofiadm -d /dev/lofi/1\fR
602 # \fBlofiadm -U /export/home/solaris.iso\fR
603 .fi
604 .in -2
605 .sp

607 .LP
608 \fBExample 6 \fRCreating an Encrypted UFS File System on a File
609 .sp
610 .LP
611 This example is similar to the example of making a UFS filesystem on a file,
612 above.

614 .sp
615 .LP
616 Create the file:

618 .sp
619 .in +2
620 .nf
621 # \fBmkfile 35m /export/home/test\fR
622 .fi
623 .in -2
624 .sp

626 .sp
627 .LP
628 Attach the file to a block device and specify that the file image is encrypted.
629 As a result of this command, you obtain the character device, which is
630 subsequently used by \fBnewfs\fR:

632 .sp
633 .in +2
634 .nf
635 # \fBlofiadm -c aes-256-cbc -a /export/home/secrets\fR
636 Enter passphrase: \fBMy-M0th3r!l0v3s_m3+4lw4ys!\fR          (\fBnot echoed\fR)
637 Re-enter passphrase: \fBMy-M0th3r!l0v3s_m3+4lw4ys!\fR      (\fBnot echoed\fR)
638 /dev/lofi/1

640 # \fBnewfs /dev/rlofi/1\fR
641 newfs: construct a new file system /dev/rlofi/1: (y/n)? \fBy\fR
642 /dev/rlofi/1: 71638 sectors in 119 cylinders of 1 tracks, 602 sectors
643 35.0MB in 8 cyl groups (16 c/g, 4.70MB/g, 2240 i/g)
644 super-block backups (for fsck -F ufs -o b=#) at:
645 32, 9664, 19296, 28928, 38560, 48192, 57824, 67456,
646 .fi
647 .in -2
648 .sp

```



```

650 .sp
651 .LP
652 The mapped file system shows that encryption is enabled:

654 .sp
655 .in +2
656 .nf
657 # \fBlofiadm\fR
658 Block Device      File      Options
659 /dev/lofi/1      /export/home/secrets  Encrypted
660 .fi
661 .in -2
662 .sp

664 .sp
665 .LP
666 Mount and use the filesystem:

668 .sp
669 .in +2
670 .nf
671 # \fBmount /dev/lofi/1 /mnt\fR
672 # \fBcp moms_secret_*_recipe /mnt\fR
673 # \fBls /mnt\fR
674 \&./      moms_secret_cookie_recipe  moms_secret_soup_recipe
675 \&../      moms_secret_fudge_recipe   moms_secret_stuffing_recipe
676 lost+found/ moms_secret_meatloaf_recipe moms_secret_waffle_recipe
677 # \fBmount /mnt\fR
678 # \fBlofiadm -d /dev/lofi/1\fR
679 .fi
680 .in -2
681 .sp

683 .sp
684 .LP
685 Subsequent attempts to map the filesystem with the wrong key or the wrong
686 encryption algorithm will fail:

688 .sp
689 .in +2
690 .nf
691 # \fBlofiadm -c blowfish-cbc -a /export/home/secrets\fR
692 Enter passphrase: \fBmommy\fR      (\fInot echoed\fR)
693 Re-enter passphrase: \fBmommy\fR   (\fInot echoed\fR)
694 lofiadm: could not map file /root/lofi: Invalid argument
695 # \fBlofiadm\fR
696 Block Device      File      Options
697 #
698 .fi
699 .in -2
700 .sp

702 .sp
703 .LP
704 Attempts to map the filesystem without encryption will succeed, however
705 attempts to mount and use the filesystem will fail:

707 .sp
708 .in +2
709 .nf
710 # \fBlofiadm -a /export/home/secrets\fR
711 /dev/lofi/1
712 # \fBlofiadm\fR
713 Block Device      File      Options
714 /dev/lofi/1      /export/home/secrets  -
715 # \fBmount /dev/lofi/1 /mnt\fR

```

```

716 mount: /dev/lofi/1 is not this fstype
717 #
718 .fi
719 .in -2
720 .sp

722 .SH ENVIRONMENT VARIABLES
723 .sp
724 .LP
725 See \fBenvron\fR(5) for descriptions of the following environment variables
726 that affect the execution of \fBlofiadm\fR: \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR
727 and \fBNLS_PATH\fR.
728 .SH EXIT STATUS
729 .sp
730 .LP
731 The following exit values are returned:
732 .sp
733 .ne 2
734 .na
735 \fB0\fR
736 .ad
737 .sp .6
738 .RS 4n
739 Successful completion.
740 .RE

742 .sp
743 .ne 2
744 .na
745 \fB>0\fR
746 .ad
747 .sp .6
748 .RS 4n
749 An error occurred.
750 .RE

752 .SH SEE ALSO
753 .sp
754 .LP
755 \fBfsck\fR(1M), \fBmount\fR(1M), \fBmount_ufs\fR(1M), \fBnewfs\fR(1M),
756 \fBattributes\fR(5), \fBlofi\fR(7D), \fBflows\fR(7FS)
757 .SH NOTES
758 .sp
759 .LP
760 Just as you would not directly access a disk device that has mounted file
761 systems, you should not access a file associated with a block device except
762 through the \fBlofi\fR file driver. It might also be appropriate to ensure that
763 the file has appropriate permissions to prevent such access.
764 .sp
765 .LP
766 The abilities of \fBlofiadm\fR, and who can use them, are controlled by the
767 permissions of \fB/dev/lofictl\fR. Read-access allows query operations, such as
768 listing all the associations. Write-access is required to do any state-changing
769 operations, like adding an association. As shipped, \fB/dev/lofictl\fR is owned
770 by \fBroot\fR, in group \fBsys\fR, and mode \fB0644\fR, so all users can do
771 query operations but only root can change anything. The administrator can give
772 users write-access, allowing them to add or delete associations, but that is
773 very likely a security hole and should probably only be given to a trusted
774 group.
775 .sp
776 .LP
777 When mounting a filesystem image, take care to use appropriate mount options.
778 In particular, the \fBnosuid\fR mount option might be appropriate for \fBUBFS\fR
779 images whose origin is unknown. Also, some options might not be useful or
780 appropriate, like \fBlogging\fR or \fBforcedirectio\fR for \fBUBFS\fR. For
781 compatibility purposes, a raw device is also exported along with the block

```

new/usr/src/man/man1m/lofiadm.1m

13

782 device. For example, \fBnewfs\fR(1M) requires one.
783 .sp
784 .LP
785 The output of \fBlofiadm\fR (without arguments) might change in future
786 releases.

new/usr/src/man/man1m/root_archive.1m

1

```
*****
2466 Sat Jul 19 19:24:35 2014
new/usr/src/man/man1m/root_archive.1m
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
_____unchanged_portion_omitted_____
```

43446 Sat Jul 19 19:24:35 2014

new/usr/src/man/man1m/zonecfg.1m

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1  \" te
2  .\" Copyright (c) 2004, 2009 Sun Microsystems, Inc. All Rights Reserved.
3  .\" Copyright 2013 Joyent, Inc. All Rights Reserved.
4  .\" The contents of this file are subject to the terms of the Common Development
5  .\" See the License for the specific language governing permissions and limitati
6  .\" fields enclosed by brackets \"[]\" replaced with your own identifying informat
7  .TH ZONECFG 1M \"Feb 28, 2014\"
8  .SH NAME
9  zonecfg \- set up zone configuration
10 .SH SYNOPSIS
11 .LP
12 .nf
13 \fBzonecfg\fR \fB-z\fR \fIzonename\fR
14 .fi

16 .LP
17 .nf
18 \fBzonecfg\fR \fB-z\fR \fIzonename\fR \fIsubcommand\fR
19 .fi

21 .LP
22 .nf
23 \fBzonecfg\fR \fB-z\fR \fIzonename\fR \fB-f\fR \fIcommand_file\fR
24 .fi

26 .LP
27 .nf
28 \fBzonecfg\fR help
29 .fi

31 .SH DESCRIPTION
32 .sp
33 .LP
34 The \fBzonecfg\fR utility creates and modifies the configuration of a zone.
35 Zone configuration consists of a number of resources and properties.
36 .sp
37 .LP
38 To simplify the user interface, \fBzonecfg\fR uses the concept of a scope. The
39 default scope is global.
40 .sp
41 .LP
42 The following synopsis of the \fBzonecfg\fR command is for interactive usage:
43 .sp
44 .in +2
45 .nf
46 zonecfg \fB-z\fR \fIzonename subcommand\fR
47 .fi
48 .in -2
49 .sp

51 .sp
52 .LP
53 Parameters changed through \fBzonecfg\fR do not affect a running zone. The zone
54 must be rebooted for the changes to take effect.
55 .sp
56 .LP

```

```

57 In addition to creating and modifying a zone, the \fBzonecfg\fR utility can
58 also be used to persistently specify the resource management settings for the
59 global zone.
60 .sp
61 .LP
62 In the following text, "rctl" is used as an abbreviation for "resource
63 control". See \fBresource_controls\fR(5).
64 .sp
65 .LP
66 Every zone is configured with an associated brand. The brand determines the
67 user-level environment used within the zone, as well as various behaviors for
68 the zone when it is installed, boots, or is shutdown. Once a zone has been
69 installed the brand cannot be changed. The default brand is determined by the
70 installed distribution in the global zone. Some brands do not support all of
71 the \fBzonecfg\fR properties and resources. See the brand-specific man page for
72 more details on each brand. For an overview of brands, see the \fBbrands\fR(5)
73 man page.
74 .SS "Resources"
75 .sp
76 .LP
77 The following resource types are supported:
78 .sp
79 .ne 2
80 .na
81 \fB\fBattr\fR\fR
82 .ad
83 .sp .6
84 .RS 4n
85 Generic attribute.
86 .RE

88 .sp
89 .ne 2
90 .na
91 \fB\fBcapped-cpu\fR\fR
92 .ad
93 .sp .6
94 .RS 4n
95 Limits for CPU usage.
96 .RE

98 .sp
99 .ne 2
100 .na
101 \fB\fBcapped-memory\fR\fR
102 .ad
103 .sp .6
104 .RS 4n
105 Limits for physical, swap, and locked memory.
106 .RE

108 .sp
109 .ne 2
110 .na
111 \fB\fBdataset\fR\fR
112 .ad
113 .sp .6
114 .RS 4n
115 \fBZFS\fR dataset.
116 .RE

118 .sp
119 .ne 2
120 .na
121 \fB\fBdedicated-cpu\fR\fR
122 .ad

```

```

123 .sp .6
124 .RS 4n
125 Subset of the system's processors dedicated to this zone while it is running.
126 .RE

128 .sp
129 .ne 2
130 .na
131 \fB\fBdevice\fR\fR
132 .ad
133 .sp .6
134 .RS 4n
135 Device.
136 .RE

138 .sp
139 .ne 2
140 .na
141 \fB\fBfs\fR\fR
142 .ad
143 .sp .6
144 .RS 4n
145 file-system
146 .RE

148 .sp
149 .ne 2
150 .na
151 \fB\fBnet\fR\fR
152 .ad
153 .sp .6
154 .RS 4n
155 Network interface.
156 .RE

158 .sp
159 .ne 2
160 .na
161 \fB\fBrcctl\fR\fR
162 .ad
163 .sp .6
164 .RS 4n
165 Resource control.
166 .RE

168 .SS "Properties"
169 .sp
170 .LP
171 Each resource type has one or more properties. There are also some global
172 properties, that is, properties of the configuration as a whole, rather than of
173 some particular resource.
174 .sp
175 .LP
176 The following properties are supported:
177 .sp
178 .ne 2
179 .na
180 \fB(global)\fR
181 .ad
182 .sp .6
183 .RS 4n
184 \fB\fbzonename\fR
185 .RE

187 .sp
188 .ne 2

```

```

189 .na
190 \fB(global)\fR
191 .ad
192 .sp .6
193 .RS 4n
194 \fB\fbzonepath\fR
195 .RE

197 .sp
198 .ne 2
199 .na
200 \fB(global)\fR
201 .ad
202 .sp .6
203 .RS 4n
204 \fB\fbautoboot\fR
205 .RE

207 .sp
208 .ne 2
209 .na
210 \fB(global)\fR
211 .ad
212 .sp .6
213 .RS 4n
214 \fB\fbbootargs\fR
215 .RE

217 .sp
218 .ne 2
219 .na
220 \fB(global)\fR
221 .ad
222 .sp .6
223 .RS 4n
224 \fB\fbpool\fR
225 .RE

227 .sp
228 .ne 2
229 .na
230 \fB(global)\fR
231 .ad
232 .sp .6
233 .RS 4n
234 \fB\fblimitpriv\fR
235 .RE

237 .sp
238 .ne 2
239 .na
240 \fB(global)\fR
241 .ad
242 .sp .6
243 .RS 4n
244 \fB\fbbrand\fR
245 .RE

247 .sp
248 .ne 2
249 .na
250 \fB(global)\fR
251 .ad
252 .sp .6
253 .RS 4n
254 \fB\fbcpu-shares\fR

```

```

255 .RE
257 .sp
258 .ne 2
259 .na
260 \fB(global)\fR
261 .ad
262 .sp .6
263 .RS 4n
264 \fBhostid\fR
265 .RE
267 .sp
268 .ne 2
269 .na
270 \fB(global)\fR
271 .ad
272 .sp .6
273 .RS 4n
274 \fBmax-lwps\fR
275 .RE
277 .sp
278 .ne 2
279 .na
280 \fB(global)\fR
281 .ad
282 .sp .6
283 .RS 4n
284 \fBmax-msg-ids\fR
285 .RE
287 .sp
288 .ne 2
289 .na
290 \fB(global)\fR
291 .ad
292 .sp .6
293 .RS 4n
294 \fBmax-sem-ids\fR
295 .RE
297 .sp
298 .ne 2
299 .na
300 \fB(global)\fR
301 .ad
302 .sp .6
303 .RS 4n
304 \fBmax-shm-ids\fR
305 .RE
307 .sp
308 .ne 2
309 .na
310 \fB(global)\fR
311 .ad
312 .sp .6
313 .RS 4n
314 \fBmax-shm-memory\fR
315 .RE
317 .sp
318 .ne 2
319 .na
320 \fB(global)\fR

```

```

321 .ad
322 .sp .6
323 .RS 4n
324 \fBscheduling-class\fR
325 .RE
327 .sp
328 .ne 2
329 .na
330 .B (global)
331 .ad
332 .sp .6
333 .RS 4n
334 .B fs-allowed
335 .RE
337 .sp
338 .ne 2
339 .na
340 \fB\fBs\fR\fR
341 .ad
342 .sp .6
343 .RS 4n
344 \fBdir\fR, \fBspecial\fR, \fBraw\fR, \fBtype\fR, \fBoptions\fR
345 .RE
347 .sp
348 .ne 2
349 .na
350 \fB\fBnet\fR\fR
351 .ad
352 .sp .6
353 .RS 4n
354 \fBaddress\fR, \fBphysical\fR, \fBdefrouter\fR
355 .RE
357 .sp
358 .ne 2
359 .na
360 \fB\fBdevice\fR\fR
361 .ad
362 .sp .6
363 .RS 4n
364 \fBmatch\fR
365 .RE
367 .sp
368 .ne 2
369 .na
370 \fB\fBrcctl\fR\fR
371 .ad
372 .sp .6
373 .RS 4n
374 \fBname\fR, \fBvalue\fR
375 .RE
377 .sp
378 .ne 2
379 .na
380 \fB\fBattr\fR\fR
381 .ad
382 .sp .6
383 .RS 4n
384 \fBname\fR, \fBtype\fR, \fBvalue\fR
385 .RE

```

```

387 .sp
388 .ne 2
389 .na
390 \fB\fBdataset\fR\fR
391 .ad
392 .sp .6
393 .RS 4n
394 \fBname\fR
395 .RE

397 .sp
398 .ne 2
399 .na
400 \fB\fBdedicated-cpu\fR\fR
401 .ad
402 .sp .6
403 .RS 4n
404 \fBncpus\fR, \fBimportance\fR
405 .RE

407 .sp
408 .ne 2
409 .na
410 \fB\fBcapped-memory\fR\fR
411 .ad
412 .sp .6
413 .RS 4n
414 \fBphysical\fR, \fBswap\fR, \fBblocked\fR
415 .RE

417 .sp
418 .ne 2
419 .na
420 \fB\fBcapped-cpu\fR\fR
421 .ad
422 .sp .6
423 .RS 4n
424 \fBncpus\fR
425 .RE

427 .sp
428 .LP
429 As for the property values which are paired with these names, they are either
430 simple, complex, or lists. The type allowed is property-specific. Simple values
431 are strings, optionally enclosed within quotation marks. Complex values have
432 the syntax:
433 .sp
434 .in +2
435 .nf
436 (<\fIname\fR>=<\fIvalue\fR>,<\fIname\fR>=<\fIvalue\fR>,...)
437 .fi
438 .in -2
439 .sp

441 .sp
442 .LP
443 where each <\fIvalue\fR> is simple, and the <\fIname\fR> strings are unique
444 within a given property. Lists have the syntax:
445 .sp
446 .in +2
447 .nf
448 [<\fIvalue\fR>,...]
449 .fi
450 .in -2
451 .sp

```

```

453 .sp
454 .LP
455 where each <\fIvalue\fR> is either simple or complex. A list of a single value
456 (either simple or complex) is equivalent to specifying that value without the
457 list syntax. That is, "foo" is equivalent to "[foo]". A list can be empty
458 (denoted by "[ ]").
459 .sp
460 .LP
461 In interpreting property values, \fBzonecfg\fR accepts regular expressions as
462 specified in \fBfnmatch\fR(5). See \fBEXAMPLES\fR.
463 .sp
464 .LP
465 The property types are described as follows:
466 .sp
467 .ne 2
468 .na
469 \fBglobal: \fBzonename\fR\fR
470 .ad
471 .sp .6
472 .RS 4n
473 The name of the zone.
474 .RE

476 .sp
477 .ne 2
478 .na
479 \fBglobal: \fBzonepath\fR\fR
480 .ad
481 .sp .6
482 .RS 4n
483 Path to zone's file system.
484 .RE

486 .sp
487 .ne 2
488 .na
489 \fBglobal: \fBautoboot\fR\fR
490 .ad
491 .sp .6
492 .RS 4n
493 Boolean indicating that a zone should be booted automatically at system boot.
494 Note that if the zones service is disabled, the zone will not autoboot,
495 regardless of the setting of this property. You enable the zones service with a
496 \fBsvcadm\fR command, such as:
497 .sp
498 .in +2
499 .nf
500 # \fBsvcadm enable svc:/system/zones:default\fR
501 .fi
502 .in -2
503 .sp

505 Replace \fBenable\fR with \fBdisable\fR to disable the zones service. See
506 \fBsvcadm\fR(1M).
507 .RE

509 .sp
510 .ne 2
511 .na
512 \fBglobal: \fBbootargs\fR\fR
513 .ad
514 .sp .6
515 .RS 4n
516 Arguments (options) to be passed to the zone bootup, unless options are
517 supplied to the "\fBzoneadm boot\fR" command, in which case those take
518 precedence. The valid arguments are described in \fBzoneadm\fR(1M).

```

```

519 .RE
521 .sp
522 .ne 2
523 .na
524 \fBglobal: \fBpool\fR\fR
525 .ad
526 .sp .6
527 .RS 4n
528 Name of the resource pool that this zone must be bound to when booted. This
529 property is incompatible with the \fBdedicated-cpu\fR resource.
530 .RE
532 .sp
533 .ne 2
534 .na
535 \fBglobal: \fBlimitpriv\fR\fR
536 .ad
537 .sp .6
538 .RS 4n
539 The maximum set of privileges any process in this zone can obtain. The property
540 should consist of a comma-separated privilege set specification as described in
541 \fBpriv_str_to_set\fR(3C). Privileges can be excluded from the resulting set by
542 preceding their names with a dash (-) or an exclamation point (!). The special
543 privilege string "zone" is not supported in this context. If the special string
544 "default" occurs as the first token in the property, it expands into a safe set
545 of privileges that preserve the resource and security isolation described in
546 \fBzones\fR(5). A missing or empty property is equivalent to this same set of
547 safe privileges.
548 .sp
549 The system administrator must take extreme care when configuring privileges for
550 a zone. Some privileges cannot be excluded through this mechanism as they are
551 required in order to boot a zone. In addition, there are certain privileges
552 which cannot be given to a zone as doing so would allow processes inside a zone
553 to unduly affect processes in other zones. \fBzoneadm\fR(1M) indicates when an
554 invalid privilege has been added or removed from a zone's privilege set when an
555 attempt is made to either "boot" or "ready" the zone.
556 .sp
557 See \fBprivileges\fR(5) for a description of privileges. The command "\fBppriv
558 -l\fR" (see \fBppriv\fR(1)) produces a list of all Solaris privileges. You can
559 specify privileges as they are displayed by \fBppriv\fR. In
560 \fBprivileges\fR(5), privileges are listed in the form
561 PRIV_\fIprivilege_name\fR. For example, the privilege \fIisys_time\fR, as you
562 would specify it in this property, is listed in \fBprivileges\fR(5) as
563 \fBPRIV_SYS_TIME\fR.
564 .RE
566 .sp
567 .ne 2
568 .na
569 \fBglobal: \fBbrand\fR\fR
570 .ad
571 .sp .6
572 .RS 4n
573 The zone's brand type.
574 .RE
576 .sp
577 .ne 2
578 .na
579 \fBglobal: \fBip-type\fR\fR
580 .ad
581 .sp .6
582 .RS 4n
583 A zone can either share the IP instance with the global zone, which is the
584 default, or have its own exclusive instance of IP.

```

```

585 .sp
586 This property takes the values \fBshared\fR and \fBexclusive\fR.
587 .RE
589 .sp
590 .ne 2
591 .na
592 \fBglobal: \fBhostid\fR\fR
593 .ad
594 .sp .6
595 .RS 4n
596 A zone can emulate a 32-bit host identifier to ease system consolidation. A
597 zone's \fBhostid\fR property is empty by default, meaning that the zone does
598 not emulate a host identifier. Zone host identifiers must be hexadecimal values
599 between 0 and FFFFFFFE. A \fB0x\fR or \fB0X\fR prefix is optional. Both
600 uppercase and lowercase hexadecimal digits are acceptable.
601 .RE
603 .sp
604 .ne 2
605 .na
606 \fBfbfs\fR: dir, special, raw, type, options\fR
607 .ad
608 .sp .6
609 .RS 4n
610 Values needed to determine how, where, and so forth to mount file systems. See
611 \fBmount\fR(1M), \fBmount\fR(2), \fBfsck\fR(1M), and \fBfstab\fR(4).
612 .RE
614 .sp
615 .ne 2
616 .na
617 \fBfbnet\fR: address, physical, defrouter\fR
618 .ad
619 .sp .6
620 .RS 4n
621 The network address and physical interface name of the network interface. The
622 network address is one of:
623 .RS +4
624 .TP
625 .ie t \(\bu
626 .el o
627 a valid IPv4 address, optionally followed by "\fB/\fR" and a prefix length;
628 .RE
629 .RS +4
630 .TP
631 .ie t \(\bu
632 .el o
633 a valid IPv6 address, which must be followed by "\fB/\fR" and a prefix length;
634 .RE
635 .RS +4
636 .TP
637 .ie t \(\bu
638 .el o
639 a host name which resolves to an IPv4 address.
640 .RE
641 Note that host names that resolve to IPv6 addresses are not supported.
642 .sp
643 The physical interface name is the network interface name.
644 .sp
645 The default router is specified similarly to the network address except that it
646 must not be followed by a \fB/\fR (slash) and a network prefix length.
647 .sp
648 A zone can be configured to be either exclusive-IP or shared-IP. For a
649 shared-IP zone, you must set both the physical and address properties; setting
650 the default router is optional. The interface specified in the physical

```


651 property must be plumbed in the global zone prior to booting the non-global
652 zone. However, if the interface is not used by the global zone, it should be
653 configured `\fBdown\fR` in the global zone, and the default router for the
654 interface should be specified here.

655 .sp
656 For an exclusive-IP zone, the physical property must be set and the address and
657 default router properties cannot be set.

658 .RE

660 .sp
661 .ne 2
662 .na
663 `\fB\fBdevice\fR: match\fR`
664 .ad
665 .sp .6
666 .RS 4n
667 Device name to match.
668 .RE

670 .sp
671 .ne 2
672 .na
673 `\fB\fBrcctl\fR: name, value\fR`
674 .ad
675 .sp .6
676 .RS 4n
677 The name and `\fBipriv\fR`/`\fBilimit\fR`/`\fBiaction\fR` triple of a resource control.
678 See `\fBprctl\fR(1)` and `\fBrcctladm\fR(1M)`. The preferred way to set `rctl` values
679 is to use the global property name associated with a specific `rctl`.
680 .RE

682 .sp
683 .ne 2
684 .na
685 `\fB\fBattr\fR: name, type, value\fR`
686 .ad
687 .sp .6
688 .RS 4n
689 The name, type and value of a generic attribute. The `\fBtype\fR` must be one of
690 `\fBint\fR`, `\fBuint\fR`, `\fBboolean\fR` or `\fBstring\fR`, and the value must be of
691 that type. `\fBuint\fR` means unsigned, that is, a non-negative integer.
692 .RE

694 .sp
695 .ne 2
696 .na
697 `\fB\fBdataset\fR: name\fR`
698 .ad
699 .sp .6
700 .RS 4n
701 The name of a `\fBZFS\fR` dataset to be accessed from within the zone. See
702 `\fBzfs\fR(1M)`.
703 .RE

705 .sp
706 .ne 2
707 .na
708 `\fBglobal: \fBcpu-shares\fR`
709 .ad
710 .sp .6
711 .RS 4n
712 The number of Fair Share Scheduler (FSS) shares to allocate to this zone. This
713 property is incompatible with the `\fBdedicated-cpu\fR` resource. This property
714 is the preferred way to set the `\fBzone.cpu-shares\fR rctl`.
715 .RE

717 .sp
718 .ne 2
719 .na
720 `\fBglobal: \fBmax-lwps\fR`
721 .ad
722 .sp .6
723 .RS 4n
724 The maximum number of LWPs simultaneously available to this zone. This property
725 is the preferred way to set the `\fBzone.max-lwps\fR rctl`.
726 .RE

728 .sp
729 .ne 2
730 .na
731 `\fBglobal: \fBmax-msg-ids\fR`
732 .ad
733 .sp .6
734 .RS 4n
735 The maximum number of message queue IDs allowed for this zone. This property is
736 the preferred way to set the `\fBzone.max-msg-ids\fR rctl`.
737 .RE

739 .sp
740 .ne 2
741 .na
742 `\fBglobal: \fBmax-sem-ids\fR`
743 .ad
744 .sp .6
745 .RS 4n
746 The maximum number of semaphore IDs allowed for this zone. This property is the
747 preferred way to set the `\fBzone.max-sem-ids\fR rctl`.
748 .RE

750 .sp
751 .ne 2
752 .na
753 `\fBglobal: \fBmax-shm-ids\fR`
754 .ad
755 .sp .6
756 .RS 4n
757 The maximum number of shared memory IDs allowed for this zone. This property is
758 the preferred way to set the `\fBzone.max-shm-ids\fR rctl`.
759 .RE

761 .sp
762 .ne 2
763 .na
764 `\fBglobal: \fBmax-shm-memory\fR`
765 .ad
766 .sp .6
767 .RS 4n
768 The maximum amount of shared memory allowed for this zone. This property is the
769 preferred way to set the `\fBzone.max-shm-memory\fR rctl`. A scale (K, M, G, T)
770 can be applied to the value for this number (for example, 1M is one megabyte).
771 .RE

773 .sp
774 .ne 2
775 .na
776 `\fBglobal: \fBsched-class\fR`
777 .ad
778 .sp .6
779 .RS 4n
780 Specifies the scheduling class used for processes running in a zone. When this
781 property is not specified, the scheduling class is established as follows:
782 .RS +4

```

783 .TP
784 .ie t \(\bu
785 .el o
786 If the \fBcpu-shares\fR property or equivalent rctl is set, the scheduling
787 class FSS is used.
788 .RE
789 .RS +4
790 .TP
791 .ie t \(\bu
792 .el o
793 If neither \fBcpu-shares\fR nor the equivalent rctl is set and the zone's pool
794 property references a pool that has a default scheduling class, that class is
795 used.
796 .RE
797 .RS +4
798 .TP
799 .ie t \(\bu
800 .el o
801 Under any other conditions, the system default scheduling class is used.
802 .RE
803 .RE

807 .sp
808 .ne 2
809 .na
810 \fB\fBdedicated-cpu\fR: ncpus, importance\fR
811 .ad
812 .sp .6
813 .RS 4n
814 The number of CPUs that should be assigned for this zone's exclusive use. The
815 zone will create a pool and processor set when it boots. See \fBpooladm\fR(1M)
816 and \fBpoolcfg\fR(1M) for more information on resource pools. The \fBncpu\fR
817 property can specify a single value or a range (for example, 1-4) of
818 processors. The \fBimportance\fR property is optional; if set, it will specify
819 the \fBpset.importance\fR value for use by \fBpoold\fR(1M). If this resource is
820 used, there must be enough free processors to allocate to this zone when it
821 boots or the zone will not boot. The processors assigned to this zone will not
822 be available for the use of the global zone or other zones. This resource is
823 incompatible with both the \fBpool\fR and \fBcpu-shares\fR properties. Only a
824 single instance of this resource can be added to the zone.
825 .RE

827 .sp
828 .ne 2
829 .na
830 \fB\fBcapped-memory\fR: physical, swap, locked\fR
831 .ad
832 .sp .6
833 .RS 4n
834 The caps on the memory that can be used by this zone. A scale (K, M, G, T) can
835 be applied to the value for each of these numbers (for example, 1M is one
836 megabyte). Each of these properties is optional but at least one property must
837 be set when adding this resource. Only a single instance of this resource can
838 be added to the zone. The \fBphysical\fR property sets the \fBmax-rss\fR for
839 this zone. This will be enforced by \fBrcapd\fR(1M) running in the global zone.
840 The \fBswap\fR property is the preferred way to set the \fBzone.max-swap\fR
841 rctl. The \fBblocked\fR property is the preferred way to set the
842 \fBzone.max-locked-memory\fR rctl.
843 .RE

845 .sp
846 .ne 2
847 .na
848 \fB\fBcapped-cpu\fR: ncpus\fR

```

```

849 .ad
850 .sp .6
851 .RS 4n
852 Sets a limit on the amount of CPU time that can be used by a zone. The unit
853 used translates to the percentage of a single CPU that can be used by all user
854 threads in a zone, expressed as a fraction (for example, \fB%.75\fR) or a
855 mixed number (whole number and fraction, for example, \fB1.25\fR). An
856 \fBncpu\fR value of \fB1\fR means 100% of a CPU, a value of \fB1.25\fR means
857 125%, \fB%.75\fR mean 75%, and so forth. When projects within a capped zone
858 have their own caps, the minimum value takes precedence.
859 .sp
860 The \fBcapped-cpu\fR property is an alias for \fBzone.cpu-cap\fR resource
861 control and is related to the \fBzone.cpu-cap\fR resource control. See
862 \fBresource_controls\fR(5).
863 .RE

865 .sp
866 .ne 2
867 .mk
868 \fBglobal: \fBfs-allowed\fR
869 .ad
870 .sp .6
871 .RS 4n
872 A comma-separated list of additional filesystems that may be mounted within
873 the zone; for example "ufs,pcfs". By default, only hfs(7fs) and network
874 filesystems can be mounted. If the first entry in the list is "-" then
875 that disables all of the default filesystems. If any filesystems are listed
876 after "-" then only those filesystems can be mounted.

878 This property does not apply to filesystems mounted into the zone via "add fs"
879 or "add dataset".

881 WARNING: allowing filesystem mounts other than the default may allow the zone
882 administrator to compromise the system with a malicious filesystem image, and
883 is not supported.
884 .RE

886 .sp
887 .LP
888 The following table summarizes resources, property-names, and types:
889 .sp
890 .in +2
891 .nf
892 resource          property-name  type
893 (global)         zonename      simple
894 (global)         zonename      simple
895 (global)         autoboot      simple
896 (global)         bootargs      simple
897 (global)         pool          simple
898 (global)         limitpriv    simple
899 (global)         brand         simple
900 (global)         ip-type       simple
901 (global)         hostid       simple
902 (global)         cpu-shares    simple
903 (global)         max-lwps     simple
904 (global)         max-msg-ids   simple
905 (global)         max-sem-ids   simple
906 (global)         max-shm-ids   simple
907 (global)         max-shm-memory simple
908 (global)         scheduling-class simple
909 fs               dir           simple
910                 special      simple
911                 raw          simple
912                 type         simple
913                 options     list of simple

```

```

914 net          address      simple
915             physical     simple
916 device       match        simple
917 rctl         name         simple
918             value        list of complex
919 attr         name         simple
920             type         simple
921             value        simple
922 dataset      name         simple
923 dedicated-cpu ncpus       simple or range
924             importance   simple

926 capped-memory physical   simple with scale
927             swap        simple with scale
928             locked      simple with scale

930 capped-cpu   ncpus       simple
931 .fi
932 .in -2
933 .sp

935 .sp
936 .LP
937 To further specify things, the breakdown of the complex property "value" of the
938 "rctl" resource type, it consists of three name/value pairs, the names being
939 "priv", "limit" and "action", each of which takes a simple value. The "name"
940 property of an "attr" resource is syntactically restricted in a fashion similar
941 but not identical to zone names: it must begin with an alphanumeric, and can
942 contain alphanumerics plus the hyphen (\fB-\fR), underscore (\fB_\fR), and dot
943 (\fB&.\fR) characters. Attribute names beginning with "zone" are reserved for
944 use by the system. Finally, the "autoboot" global property must have a value of
945 "true" or "false".
946 .SS "Using Kernel Statistics to Monitor CPU Caps"
947 .sp
948 .LP
949 Using the kernel statistics (\fBkstat\fR(3KSTAT)) module \fBcaps\fR, the system
950 maintains information for all capped projects and zones. You can access this
951 information by reading kernel statistics (\fBkstat\fR(3KSTAT)), specifying
952 \fBcaps\fR as the \fBkstat\fR module name. The following command displays
953 kernel statistics for all active CPU caps:
954 .sp
955 .in +2
956 .nf
957 # \fBkstat caps::'/cpucaps/'\fR
958 .fi
959 .in -2
960 .sp

962 .sp
963 .LP
964 A \fBkstat\fR(1M) command running in a zone displays only CPU caps relevant for
965 that zone and for projects in that zone. See \fBEXAMPLES\fR.
966 .sp
967 .LP
968 The following are cap-related arguments for use with \fBkstat\fR(1M):
969 .sp
970 .ne 2
971 .na
972 \fB\fBcaps\fR\fR
973 .ad
974 .sp .6
975 .RS 4n
976 The \fBkstat\fR module.
977 .RE

979 .sp

```

```

980 .ne 2
981 .na
982 \fB\fBproject_caps\fR or \fBzone_caps\fR\fR
983 .ad
984 .sp .6
985 .RS 4n
986 \fBkstat\fR class, for use with the \fBkstat\fR \fB-c\fR option.
987 .RE

989 .sp
990 .ne 2
991 .na
992 \fB\fBcpucaps_project_\fR\fR or \fBcpucaps_zone_\fR\fR
993 .ad
994 .sp .6
995 .RS 4n
996 \fBkstat\fR name, for use with the \fBkstat\fR \fB-n\fR option. \fR is the
997 project or zone identifier.
998 .RE

1000 .sp
1001 .LP
1002 The following fields are displayed in response to a \fBkstat\fR(1M) command
1003 requesting statistics for all CPU caps.
1004 .sp
1005 .ne 2
1006 .na
1007 \fB\fBmodule\fR\fR
1008 .ad
1009 .sp .6
1010 .RS 4n
1011 In this usage of \fBkstat\fR, this field will have the value \fBcaps\fR.
1012 .RE

1014 .sp
1015 .ne 2
1016 .na
1017 \fB\fBname\fR\fR
1018 .ad
1019 .sp .6
1020 .RS 4n
1021 As described above, \fBcpucaps_project_\fR\fR or
1022 \fBcpucaps_zone_\fR\fR
1023 .RE

1025 .sp
1026 .ne 2
1027 .na
1028 \fB\fBabove_sec\fR\fR
1029 .ad
1030 .sp .6
1031 .RS 4n
1032 Total time, in seconds, spent above the cap.
1033 .RE

1035 .sp
1036 .ne 2
1037 .na
1038 \fB\fBbelow_sec\fR\fR
1039 .ad
1040 .sp .6
1041 .RS 4n
1042 Total time, in seconds, spent below the cap.
1043 .RE

1045 .sp

```

```

1046 .ne 2
1047 .na
1048 \fB\fBmaxusage\fR\fR
1049 .ad
1050 .sp .6
1051 .RS 4n
1052 Maximum observed CPU usage.
1053 .RE

1055 .sp
1056 .ne 2
1057 .na
1058 \fB\fBnwait\fR\fR
1059 .ad
1060 .sp .6
1061 .RS 4n
1062 Number of threads on cap wait queue.
1063 .RE

1065 .sp
1066 .ne 2
1067 .na
1068 \fB\fBusage\fR\fR
1069 .ad
1070 .sp .6
1071 .RS 4n
1072 Current aggregated CPU usage for all threads belonging to a capped project or
1073 zone, in terms of a percentage of a single CPU.
1074 .RE

1076 .sp
1077 .ne 2
1078 .na
1079 \fB\fBvalue\fR\fR
1080 .ad
1081 .sp .6
1082 .RS 4n
1083 The cap value, in terms of a percentage of a single CPU.
1084 .RE

1086 .sp
1087 .ne 2
1088 .na
1089 \fB\fBzonename\fR\fR
1090 .ad
1091 .sp .6
1092 .RS 4n
1093 Name of the zone for which statistics are displayed.
1094 .RE

1096 .sp
1097 .LP
1098 See \fBEXAMPLES\fR for sample output from a \fBkstat\fR command.
1099 .SH OPTIONS
1100 .sp
1101 .LP
1102 The following options are supported:
1103 .sp
1104 .ne 2
1105 .na
1106 \fB\fB-f\fR \fIcommand_file\fR
1107 .ad
1108 .sp .6
1109 .RS 4n
1110 Specify the name of \fBzoncfg\fR command file. \fIcommand_file\fR is a text
1111 file of \fBzoncfg\fR subcommands, one per line.

```

```

1112 .RE

1114 .sp
1115 .ne 2
1116 .na
1117 \fB\fB-z\fR \fIzonename\fR\fR
1118 .ad
1119 .sp .6
1120 .RS 4n
1121 Specify the name of a zone. Zone names are case sensitive. Zone names must
1122 begin with an alphanumeric character and can contain alphanumeric characters,
1123 the underscore (\fB_\fR) the hyphen (\fB-\fR), and the dot (\fB&.\fR). The
1124 name \fBglobal\fR and all names beginning with \fBSUNW\fR are reserved and
1125 cannot be used.
1126 .RE

1128 .SH SUBCOMMANDS
1129 .sp
1130 .LP
1131 You can use the \fBadd\fR and \fBselect\fR subcommands to select a specific
1132 resource, at which point the scope changes to that resource. The \fBend\fR and
1133 \fBcancel\fR subcommands are used to complete the resource specification, at
1134 which time the scope is reverted back to global. Certain subcommands, such as
1135 \fBadd\fR, \fBremove\fR and \fBset\fR, have different semantics in each scope.
1136 .sp
1137 .LP
1138 \fBzoncfg\fR supports a semicolon-separated list of subcommands. For example:
1139 .sp
1140 .in +2
1141 .nf
1142 # \fBzoncfg -z myzone "add net; set physical=myvnic; end"\fR
1143 .fi
1144 .in -2
1145 .sp

1147 .sp
1148 .LP
1149 Subcommands which can result in destructive actions or loss of work have an
1150 \fB-F\fR option to force the action. If input is from a terminal device, the
1151 user is prompted when appropriate if such a command is given without the
1152 \fB-F\fR option otherwise, if such a command is given without the \fB-F\fR
1153 option, the action is disallowed, with a diagnostic message written to standard
1154 error.
1155 .sp
1156 .LP
1157 The following subcommands are supported:
1158 .sp
1159 .ne 2
1160 .na
1161 \fB\fBadd\fR \fIresource-type\fR (global scope)\fR
1162 .ad
1163 .br
1164 .na
1165 \fB\fBadd\fR \fIproperty-name property-value\fR (resource scope)\fR
1166 .ad
1167 .sp .6
1168 .RS 4n
1169 In the global scope, begin the specification for a given resource type. The
1170 scope is changed to that resource type.
1171 .sp
1172 In the resource scope, add a property of the given name with the given value.
1173 The syntax for property values varies with different property types. In
1174 general, it is a simple value or a list of simple values enclosed in square
1175 brackets, separated by commas (\fB[foo,bar,baz]\fR). See \fBPROPERTIES\fR.
1176 .RE

```

```

1178 .sp
1179 .ne 2
1180 .na
1181 \fB\fBcancel\fR\fR
1182 .ad
1183 .sp .6
1184 .RS 4n
1185 End the resource specification and reset scope to global. Abandons any
1186 partially specified resources. \fBcancel\fR is only applicable in the resource
1187 scope.
1188 .RE

```

```

1190 .sp
1191 .ne 2
1192 .na
1193 \fB\fBclear\fR \fIproperty-name\fR\fR
1194 .ad
1195 .sp .6
1196 .RS 4n
1197 Clear the value for the property.
1198 .RE

```

```

1200 .sp
1201 .ne 2
1202 .na
1203 \fB\fBcommit\fR\fR
1204 .ad
1205 .sp .6
1206 .RS 4n
1207 Commit the current configuration from memory to stable storage. The
1208 configuration must be committed to be used by \fBzoneadm\fR. Until the
1209 in-memory configuration is committed, you can remove changes with the
1210 \fBbrevert\fR subcommand. The \fBcommit\fR operation is attempted automatically
1211 upon completion of a \fBzoncfg\fR session. Since a configuration must be
1212 correct to be committed, this operation automatically does a verify.
1213 .RE

```

```

1215 .sp
1216 .ne 2
1217 .na
1218 \fB\fBcreate [\fR\fB-F\fR\fB] [\fR \fB-a\fR \fIpath\fR |\fB-b\fR \fB|\fR
1219 \fB-t\fR \fItemplate\fR\fB]\fR\fR
1220 .ad
1221 .sp .6
1222 .RS 4n
1223 Create an in-memory configuration for the specified zone. Use \fBcreate\fR to
1224 begin to configure a new zone. See \fBcommit\fR for saving this to stable
1225 storage.
1226 .sp
1227 If you are overwriting an existing configuration, specify the \fB-F\fR option
1228 to force the action. Specify the \fB-t\fR \fItemplate\fR option to create a
1229 configuration identical to \fItemplate\fR, where \fItemplate\fR is the name of
1230 a configured zone.
1231 .sp
1232 Use the \fB-a\fR \fIpath\fR option to facilitate configuring a detached zone on
1233 a new host. The \fIpath\fR parameter is the zonepath location of a detached
1234 zone that has been moved on to this new host. Once the detached zone is
1235 configured, it should be installed using the "\fBzoneadm attach\fR" command
1236 (see \fBzoneadm\fR(1M)). All validation of the new zone happens during the
1237 \fBattach\fR process, not during zone configuration.
1238 .sp
1239 Use the \fB-b\fR option to create a blank configuration. Without arguments,
1240 \fBcreate\fR applies the Sun default settings.
1241 .RE

```

```

1243 .sp

```

```

1244 .ne 2
1245 .na
1246 \fB\fBdelete [\fR\fB-F\fR\fB]\fR\fR
1247 .ad
1248 .sp .6
1249 .RS 4n
1250 Delete the specified configuration from memory and stable storage. This action
1251 is instantaneous, no commit is necessary. A deleted configuration cannot be
1252 reverted.
1253 .sp
1254 Specify the \fB-F\fR option to force the action.
1255 .RE

```

```

1257 .sp
1258 .ne 2
1259 .na
1260 \fB\fBend\fR\fR
1261 .ad
1262 .sp .6
1263 .RS 4n
1264 End the resource specification. This subcommand is only applicable in the
1265 resource scope. \fBzoncfg\fR checks to make sure the current resource is
1266 completely specified. If so, it is added to the in-memory configuration (see
1267 \fBcommit\fR for saving this to stable storage) and the scope reverts to
1268 global. If the specification is incomplete, it issues an appropriate error
1269 message.
1270 .RE

```

```

1272 .sp
1273 .ne 2
1274 .na
1275 \fB\fBexport [\fR\fB-f\fR \fIoutput-file\fR\fB]\fR\fR
1276 .ad
1277 .sp .6
1278 .RS 4n
1279 Print configuration to standard output. Use the \fB-f\fR option to print the
1280 configuration to \fIoutput-file\fR. This option produces output in a form
1281 suitable for use in a command file.
1282 .RE

```

```

1284 .sp
1285 .ne 2
1286 .na
1287 \fB\fBhelp [usage] [\fR \fB-I\fR] [syntax] [\fR \fB-command-name\fR\fB]\fR\fR
1288 .ad
1289 .sp .6
1290 .RS 4n
1291 Print general help or help about given topic.
1292 .RE

```

```

1294 .sp
1295 .ne 2
1296 .na
1297 \fB\fBinfo zonename | zonepath | autoboot | brand | pool | limitpriv\fR\fR
1298 .ad
1299 .br
1300 .na
1301 \fB\fBinfo [\fR \fB-resource-type\fR
1302 \fB|\fR \fB-property-name\fR \fB=\fR \fB-property-value\fR \fB*]\fR\fR
1303 .ad
1304 .sp .6
1305 .RS 4n
1306 Display information about the current configuration. If \fB-resource-type\fR is
1307 specified, displays only information about resources of the relevant type. If
1308 any \fB-property-name\fR value pairs are specified, displays only information
1309 about resources meeting the given criteria. In the resource scope, any

```

```

1310 arguments are ignored, and \fBinfo\fR displays information about the resource
1311 which is currently being added or modified.
1312 .RE

1314 .sp
1315 .ne 2
1316 .na
1317 \fB\fBremove\fR \fIresource-type\fR \fB{\fR\fIproperty-name\fR \fB=\fR\fIproperty
1318 -value\fR \fB}\fR(global scope)\fR
1319 .ad
1320 .sp .6
1321 .RS 4n
1322 In the global scope, removes the specified resource. The \fB[]\fR syntax means
1323 0 or more of whatever is inside the square braces. If you want only to remove a
1324 single instance of the resource, you must specify enough property name-value
1325 pairs for the resource to be uniquely identified. If no property name-value
1326 pairs are specified, all instances will be removed. If there is more than one
1327 pair is specified, a confirmation is required, unless you use the \fB-F\fR
1328 option.
1329 .RE

1331 .sp
1332 .ne 2
1333 .na
1334 \fB\fBselect\fR \fIresource-type\fR
1335 \fB{\fR \fIproperty-name\fR \fB=\fR \fIproperty-value\fR \fB}\fR \fR
1336 .ad
1337 .sp .6
1338 .RS 4n
1339 Select the resource of the given type which matches the given
1340 \fIproperty-name\fR \fIproperty-value\fR pair criteria, for modification. This
1341 subcommand is applicable only in the global scope. The scope is changed to that
1342 resource type. The \fB{\fR syntax means 1 or more of whatever is inside the
1343 curly braces. You must specify enough \fIproperty -name property-value\fR pairs
1344 for the resource to be uniquely identified.
1345 .RE

1347 .sp
1348 .ne 2
1349 .na
1350 \fB\fBset\fR \fIproperty-name\fR \fB=\fR \fIproperty\fR \fB-\fR \fIvalue\fR \fR
1351 .ad
1352 .sp .6
1353 .RS 4n
1354 Set a given property name to the given value. Some properties (for example,
1355 \fBzonepath\fR and \fBzonepath\fR) are global while others are
1356 resource-specific. This subcommand is applicable in both the global and
1357 resource scopes.
1358 .RE

1360 .sp
1361 .ne 2
1362 .na
1363 \fB\fBverify\fR \fR
1364 .ad
1365 .sp .6
1366 .RS 4n
1367 Verify the current configuration for correctness:
1368 .RS +4
1369 .TP
1370 .ie t \(\bu
1371 .el o
1372 All resources have all of their required properties specified.
1373 .RE
1374 .RS +4
1375 .TP

```

```

1376 .ie t \(\bu
1377 .el o
1378 A \fBzonepath\fR is specified.
1379 .RE
1380 .RE

1382 .sp
1383 .ne 2
1384 .na
1385 \fB\fBvert\fR \fB{\fR \fB-F\fR \fB}\fR \fR
1386 .ad
1387 .sp .6
1388 .RS 4n
1389 Revert the configuration back to the last committed state. The \fB-F\fR option
1390 can be used to force the action.
1391 .RE

1393 .sp
1394 .ne 2
1395 .na
1396 \fB\fBexit [\fR \fB-F\fR \fB]\fR \fR
1397 .ad
1398 .sp .6
1399 .RS 4n
1400 Exit the \fBzoncfg\fR session. A commit is automatically attempted if needed.
1401 You can also use an \fBEOF\fR character to exit \fBzoncfg\fR. The \fB-F\fR
1402 option can be used to force the action.
1403 .RE

1405 .SH EXAMPLES
1406 .LP
1407 \fBExample 1 \fRCreating the Environment for a New Zone
1408 .sp
1409 .LP
1410 In the following example, \fBzoncfg\fR creates the environment for a new zone.
1411 \fB/usr/local\fR is loopback mounted from the global zone into
1412 \fB/opt/local\fR. \fB/opt/sfw\fR is loopback mounted from the global zone,
1413 three logical network interfaces are added, and a limit on the number of
1414 fair-share scheduler (FSS) CPU shares for a zone is set using the \fBbrctl\fR
1415 resource type. The example also shows how to select a given resource for
1416 modification.

1418 .sp
1419 .in +2
1420 .nf
1421 example# \fBzoncfg -z myzone3\fR
1422 my-zone3: No such zone configured
1423 Use 'create' to begin configuring a new zone.
1424 zoncfg:myzone3> \fBcreate\fR
1425 zoncfg:myzone3> \fBset zonepath=/export/home/my-zone3\fR
1426 zoncfg:myzone3> \fBset autoboot=true\fR
1427 zoncfg:myzone3> \fBadd fs\fR
1428 zoncfg:myzone3:fs> \fBset dir=/usr/local\fR
1429 zoncfg:myzone3:fs> \fBset special=/opt/local\fR
1430 zoncfg:myzone3:fs> \fBset type=lofs\fR
1431 zoncfg:myzone3:fs> \fBadd options [ro,nodevices]\fR
1432 zoncfg:myzone3:fs> \fBend\fR
1433 zoncfg:myzone3> \fBadd fs\fR
1434 zoncfg:myzone3:fs> \fBset dir=/mnt\fR
1435 zoncfg:myzone3:fs> \fBset special=/dev/dsk/c0t0d0s7\fR
1436 zoncfg:myzone3:fs> \fBset raw=/dev/rdsk/c0t0d0s7\fR
1437 zoncfg:myzone3:fs> \fBset type=ufs\fR
1438 zoncfg:myzone3:fs> \fBend\fR
1439 zoncfg:myzone3> \fBadd net\fR
1440 zoncfg:myzone3:net> \fBset address=192.168.0.1/24\fR
1441 zoncfg:myzone3:net> \fBset physical=eri0\fR

```

```

1442 zonecfg:myzone3:net> \fBend\fR
1443 zonecfg:myzone3> \fBadd net\fR
1444 zonecfg:myzone3:net> \fBset address=192.168.1.2/24\fR
1445 zonecfg:myzone3:net> \fBset physical=eri0\fR
1446 zonecfg:myzone3:net> \fBend\fR
1447 zonecfg:myzone3> \fBadd net\fR
1448 zonecfg:myzone3:net> \fBset address=192.168.2.3/24\fR
1449 zonecfg:myzone3:net> \fBset physical=eri0\fR
1450 zonecfg:myzone3:net> \fBend\fR
1451 zonecfg:my-zone3> \fBset cpu-shares=5\fR
1452 zonecfg:my-zone3> \fBadd capped-memory\fR
1453 zonecfg:my-zone3:capped-memory> \fBset physical=50m\fR
1454 zonecfg:my-zone3:capped-memory> \fBset swap=100m\fR
1455 zonecfg:my-zone3:capped-memory> \fBend\fR
1456 zonecfg:myzone3> \fBexit\fR
1457 .fi
1458 .in -2
1459 .sp

1461 .LP
1462 \fBExample 2 \fRCreating a Non-Native Zone
1463 .sp
1464 .LP
1465 The following example creates a new Linux zone:

1467 .sp
1468 .in +2
1469 .nf
1470 example# \fBzonecfg -z lxzone\fR
1471 lxzone: No such zone configured
1472 Use 'create' to begin configuring a new zone
1473 zonecfg:lxzone> \fBcreate -t SUNWlx\fR
1474 zonecfg:lxzone> \fBset zonepath=/export/zones/lxzone\fR
1475 zonecfg:lxzone> \fBset autoboot=true\fR
1476 zonecfg:lxzone> \fBexit\fR
1477 .fi
1478 .in -2
1479 .sp

1481 .LP
1482 \fBExample 3 \fRCreating an Exclusive-IP Zone
1483 .sp
1484 .LP
1485 The following example creates a zone that is granted exclusive access to
1486 \fBbge1\fR and \fBbge33000\fR and that is isolated at the IP layer from the
1487 other zones configured on the system.

1489 .sp
1490 .LP
1491 The IP addresses and routing is configured inside the new zone using
1492 \fBsysidtool\fR(1M).

1494 .sp
1495 .in +2
1496 .nf
1497 example# \fBzonecfg -z excl\fR
1498 excl: No such zone configured
1499 Use 'create' to begin configuring a new zone
1500 zonecfg:excl> \fBcreate\fR
1501 zonecfg:excl> \fBset zonepath=/export/zones/excl\fR
1502 zonecfg:excl> \fBset ip-type=exclusive\fR
1503 zonecfg:excl> \fBadd net\fR
1504 zonecfg:excl:net> \fBset physical=bge1\fR
1505 zonecfg:excl:net> \fBend\fR
1506 zonecfg:excl> \fBadd net\fR
1507 zonecfg:excl:net> \fBset physical=bge33000\fR

```

```

1508 zonecfg:excl:net> \fBend\fR
1509 zonecfg:excl> \fBexit\fR
1510 .fi
1511 .in -2
1512 .sp

1514 .LP
1515 \fBExample 4 \fRAssociating a Zone with a Resource Pool
1516 .sp
1517 .LP
1518 The following example shows how to associate an existing zone with an existing
1519 resource pool:

1521 .sp
1522 .in +2
1523 .nf
1524 example# \fBzonecfg -z myzone\fR
1525 zonecfg:myzone> \fBset pool=mypool\fR
1526 zonecfg:myzone> \fBexit\fR
1527 .fi
1528 .in -2
1529 .sp

1531 .sp
1532 .LP
1533 For more information about resource pools, see \fBpooladm\fR(1M) and
1534 \fBpoolcfg\fR(1M).

1536 .LP
1537 \fBExample 5 \fRChanging the Name of a Zone
1538 .sp
1539 .LP
1540 The following example shows how to change the name of an existing zone:

1542 .sp
1543 .in +2
1544 .nf
1545 example# \fBzonecfg -z myzone\fR
1546 zonecfg:myzone> \fBset zonename=myzone2\fR
1547 zonecfg:myzone2> \fBexit\fR
1548 .fi
1549 .in -2
1550 .sp

1552 .LP
1553 \fBExample 6 \fRChanging the Privilege Set of a Zone
1554 .sp
1555 .LP
1556 The following example shows how to change the set of privileges an existing
1557 zone's processes will be limited to the next time the zone is booted. In this
1558 particular case, the privilege set will be the standard safe set of privileges
1559 a zone normally has along with the privilege to change the system date and
1560 time:

1562 .sp
1563 .in +2
1564 .nf
1565 example# \fBzonecfg -z myzone\fR
1566 zonecfg:myzone> \fBset limitpriv="default,sys_time"\fR
1567 zonecfg:myzone2> \fBexit\fR
1568 .fi
1569 .in -2
1570 .sp

1572 .LP
1573 \fBExample 7 \fRSetting the \fBzone.cpu-shares\fR Property for the Global Zone

```

```

1574 .sp
1575 .LP
1576 The following command sets the \fBzone.cpu-shares\fR property for the global
1577 zone:

1579 .sp
1580 .in +2
1581 .nf
1582 example# \fBzonecfg -z global\fR
1583 zonecfg:global> \fBset cpu-shares=5\fR
1584 zonecfg:global> \fBexit\fR
1585 .fi
1586 .in -2
1587 .sp

1589 .LP
1590 \fBExample 8 \fRUsing Pattern Matching
1591 .sp
1592 .LP
1593 The following commands illustrate \fBzonecfg\fR support for pattern matching.
1594 In the zone \fBflexlm\fR, enter:

1596 .sp
1597 .in +2
1598 .nf
1599 zonecfg:flexlm> \fBadd device\fR
1600 zonecfg:flexlm:device> \fBset match="/dev/cua/a00[2-5]"\fR
1601 zonecfg:flexlm:device> \fBend\fR
1602 .fi
1603 .in -2
1604 .sp

1606 .sp
1607 .LP
1608 In the global zone, enter:

1610 .sp
1611 .in +2
1612 .nf
1613 global# \fBls /dev/cua\fR
1614 a      a000 a001 a002 a003 a004 a005 a006 a007 b
1615 .fi
1616 .in -2
1617 .sp

1619 .sp
1620 .LP
1621 In the zone \fBflexlm\fR, enter:

1623 .sp
1624 .in +2
1625 .nf
1626 flexlm# \fBls /dev/cua\fR
1627 a002 a003 a004 a005
1628 .fi
1629 .in -2
1630 .sp

1632 .LP
1633 \fBExample 9 \fRSetting a Cap for a Zone to Three CPUs
1634 .sp
1635 .LP
1636 The following sequence uses the \fBzonecfg\fR command to set the CPU cap for a
1637 zone to three CPUs.

1639 .sp

```

```

1640 .in +2
1641 .nf
1642 zonecfg:myzone> \fBadd capped-cpu\fR
1643 zonecfg:myzone>capped-cpu> \fBset ncpus=3\fR
1644 zonecfg:myzone>capped-cpu>capped-cpu> \fBend\fR
1645 .fi
1646 .in -2
1647 .sp

1649 .sp
1650 .LP
1651 The preceding sequence, which uses the capped-cpu property, is equivalent to
1652 the following sequence, which makes use of the \fBzone.cpu-cap\fR resource
1653 control.

1655 .sp
1656 .in +2
1657 .nf
1658 zonecfg:myzone> \fBadd rctl\fR
1659 zonecfg:myzone:rctl> \fBset name=zone.cpu-cap\fR
1660 zonecfg:myzone:rctl> \fBadd value (priv=privileged,limit=300,action=none)\fR
1661 zonecfg:myzone:rctl> \fBend\fR
1662 .fi
1663 .in -2
1664 .sp

1666 .LP
1667 \fBExample 10 \fRUsing \fBkstat\fR to Monitor CPU Caps
1668 .sp
1669 .LP
1670 The following command displays information about all CPU caps.

1672 .sp
1673 .in +2
1674 .nf
1675 # \fBkstat -n /cpucaps/\fR
1676 module: caps                               instance: 0
1677 name:   cpucaps_project_0                  class:   project_caps
1678        above_sec                           0
1679        below_sec                            2157
1680        crtime                               821.048183159
1681        maxusage                             2
1682        nwait                                0
1683        snaptime                             235885.637253027
1684        usage                                0
1685        value                                18446743151372347932
1686        zonename                            global

1688 module: caps                               instance: 0
1689 name:   cpucaps_project_1                  class:   project_caps
1690        above_sec                           0
1691        below_sec                            0
1692        crtime                               225339.192787265
1693        maxusage                             5
1694        nwait                                0
1695        snaptime                             235885.637591677
1696        usage                                5
1697        value                                18446743151372347932
1698        zonename                            global

1700 module: caps                               instance: 0
1701 name:   cpucaps_project_201                class:   project_caps
1702        above_sec                           0
1703        below_sec                            235105
1704        crtime                               780.37961782
1705        maxusage                             100

```



```

1706          nwait                0
1707          snaptime              235885.637789687
1708          usage                 43
1709          value                 100
1710          zonename              global

1712 module: caps                  instance: 0
1713 name:      cpucaps_project_202 class:      project_caps
1714          above_sec             0
1715          below_sec             235094
1716          crtime               791.72983782
1717          maxusage              100
1718          nwait                 0
1719          snaptime              235885.637967512
1720          usage                 48
1721          value                 100
1722          zonename              global

1724 module: caps                  instance: 0
1725 name:      cpucaps_project_203 class:      project_caps
1726          above_sec             0
1727          below_sec             235034
1728          crtime               852.104401481
1729          maxusage              75
1730          nwait                 0
1731          snaptime              235885.638144304
1732          usage                 47
1733          value                 100
1734          zonename              global

1736 module: caps                  instance: 0
1737 name:      cpucaps_project_86710 class:      project_caps
1738          above_sec             22
1739          below_sec             235166
1740          crtime               698.441717859
1741          maxusage              101
1742          nwait                 0
1743          snaptime              235885.638319871
1744          usage                 54
1745          value                 100
1746          zonename              global

1748 module: caps                  instance: 0
1749 name:      cpucaps_zone_0      class:      zone_caps
1750          above_sec             100733
1751          below_sec             134332
1752          crtime               821.048177123
1753          maxusage              207
1754          nwait                 2
1755          snaptime              235885.638497731
1756          usage                 199
1757          value                 200
1758          zonename              global

1760 module: caps                  instance: 1
1761 name:      cpucaps_project_0   class:      project_caps
1762          above_sec             0
1763          below_sec             0
1764          crtime               225360.256448422
1765          maxusage              7
1766          nwait                 0
1767          snaptime              235885.638714404
1768          usage                 7
1769          value                 18446743151372347932
1770          zonename              test_001

```

```

1772 module: caps                  instance: 1
1773 name:      cpucaps_zone_1     class:      zone_caps
1774          above_sec             2
1775          below_sec             10524
1776          crtime               225360.256440278
1777          maxusage              106
1778          nwait                 0
1779          snaptime              235885.638896443
1780          usage                 7
1781          value                 100
1782          zonename              test_001
1783 .fi
1784 .in -2
1785 .sp

1787 .LP
1788 \fBExample 11 \fRDisplaying CPU Caps for a Specific Zone or Project
1789 .sp
1790 .LP
1791 Using the \fBkstat\fR \fB-c\fR and \fB-i\fR options, you can display CPU caps
1792 for a specific zone or project, as below. The first command produces a display
1793 for a specific project, the second for the same project within zone 1.

1795 .sp
1796 .in +2
1797 .nf
1798 # \fBkstat -c project_caps\fR

1800 # \fBkstat -c project_caps -i 1\fR
1801 .fi
1802 .in -2
1803 .sp

1805 .SH EXIT STATUS
1806 .sp
1807 .LP
1808 The following exit values are returned:
1809 .sp
1810 .ne 2
1811 .na
1812 \fB0\fR
1813 .ad
1814 .sp .6
1815 .RS 4n
1816 Successful completion.
1817 .RE

1819 .sp
1820 .ne 2
1821 .na
1822 \fB1\fR
1823 .ad
1824 .sp .6
1825 .RS 4n
1826 An error occurred.
1827 .RE

1829 .sp
1830 .ne 2
1831 .na
1832 \fB2\fR
1833 .ad
1834 .sp .6
1835 .RS 4n
1836 Invalid usage.
1837 .RE

```

```
1839 .SH ATTRIBUTES
1840 .sp
1841 .LP
1842 See \fBattributes\fR(5) for descriptions of the following attributes:
1843 .sp

1845 .sp
1846 .TS
1847 box;
1848 c | c
1849 l | l .
1850 ATTRIBUTE TYPE ATTRIBUTE VALUE
1851 _
1852 Interface Stability Volatile
1853 .TE

1855 .SH SEE ALSO
1856 .sp
1857 .LP
1858 \fBppriv\fR(1), \fBprctl\fR(1), \fBzlogin\fR(1), \fBkstat\fR(1M),
1859 \fBmount\fR(1M), \fBpooladm\fR(1M), \fBpoolcfg\fR(1M), \fBpoold\fR(1M),
1860 \fBrcapd\fR(1M), \fBrcldm\fR(1M), \fBsvcadm\fR(1M), \fBsysidtool\fR(1M),
1861 \fBzfs\fR(1M), \fBzoneadm\fR(1M), \fBpriv_str_to_set\fR(3C),
1862 \fBkstat\fR(3KSTAT), \fBvfstab\fR(4), \fBattributes\fR(5), \fBbrands\fR(5),
1863 \fBfnmatch\fR(5), \fBlx\fR(5), \fBprivileges\fR(5), \fBresource_controls\fR(5),
1864 \fBzones\fR(5)
1865 .sp
1866 .LP
1867 \fISystem Administration Guide: Solaris Containers-Resource Management, and
1868 Solaris Zones\fR
1869 .SH NOTES
1870 .sp
1871 .LP
1872 All character data used by \fBzonecfg\fR must be in US-ASCII encoding.
```

5997 Sat Jul 19 19:24:35 2014

new/usr/src/man/man2/pipe.2

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 \" te
2.\" Copyright (c) 2002, Sun Microsystems, Inc. All Rights Reserved.
3.\" Copyright 1989 AT&T
4.\" Portions Copyright (c) 2001, the Institute of Electrical and Electronics Eng
5.\" Portions Copyright (c) 2013, OmniTI Computer Consulting, Inc. All Rights Res
6.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
7.\" http://www.opengroup.org/bookstore/.
8.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
9.\" This notice shall appear on any product containing this material.
10.\" The contents of this file are subject to the terms of the Common Development
11.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
12.\" When distributing Covered Code, include this CDDL HEADER in each file and in
13.TH PIPE 2 "Apr 19, 2013"
14.SH NAME
15 pipe \- create an interprocess channel
16.SH SYNOPSIS
17.LP
18.nf
19#include <unistd.h>

21 \fBint\fR \fBpipe\fR(\fBint\fR \fBifildes\fR[2]);

23 \fBint\fR \fBpipe2\fR(\fBint\fR \fBifildes\fR[2], \fBint\fR \fBiflags\fR);
24.fi

26.SH DESCRIPTION
27.sp
28.LP
29 The \fBpipe()\fR and \fBpipe2()\fR functions create an I/O mechanism called a
30 pipe and returns two file descriptors, \fBifildes\fR[\fB0\fR] and
31 \fBifildes\fR[\fB1\fR]. The files associated with \fBifildes\fR[\fB0\fR]
32 and \fBifildes\fR[\fB1\fR] are streams and are both opened for reading and
33 writing. The \fBpipe()\fR call will clear the \fB0_NDELAY\fR,
34 \fB0_NONBLOCK\fR, and \fBFD_CLOEXEC\fR flags on both file descriptors. The
35 \fBfcntl\fR(2) function can be used to set these flags.
36.sp
37.LP
38 The \fBpipe2()\fR call will clear the \fB0_NDELAY\fR on both file descriptors.
39 The \fBiflags\fR argument may be used to specify attributes on both file
40 descriptors. \fBpipe2()\fR called with a \fBiflags\fR value of 0 will
41 behave identically to \fBpipe()\fR. Values for \fBiflags\fR are constructed
42 by a bitwise-inclusive-OR of flags from the following list, defined in
43 <\fBfcntl.h\fR>.
44.RE

45.sp
46.ne 2
47.na
48 \fB0_NONBLOCK\fR
49.ad
50.RS 12n
51 Both file descriptors will be placed in non-blocking mode. This corresponds
52 to the \fB0_NONBLOCK\fR flag to \fBfcntl\fR(2).
53.RE

55.sp
```

```

56.ne 2
57.na
58 \fB0_CLOEXEC\fR
59.ad
60.RS 12n
61 Both file descriptors will be opened with the \fB0_CLOEXEC\fR flag set. Both file
62 descriptors will be closed prior to any future \fBexec()\fR calls.
63.RE

65.sp
66.LP
67 A read from \fBifildes\fR[\fB0\fR] accesses the data written to
68 \fBifildes\fR[\fB1\fR] on a first-in-first-out (FIFO) basis and a read from
69 \fBifildes\fR[\fB1\fR] accesses the data written to \fBifildes\fR[\fB0\fR] also
70 on a \fB0_FIFO\fR basis.
71.sp
72.LP
73 Upon successful completion \fBpipe()\fR marks for update the \fBst_atime\fR,
74 \fBst_ctime\fR, and \fBst_mtime\fR fields of the pipe.
75.SH RETURN VALUES
76.sp
77.LP
78 Upon successful completion, \fB0\fR is returned. Otherwise, \fB(mil\fR is
79 returned and \fBerrno\fR is set to indicate the error.
80.SH ERRORS
81.sp
82.LP
83 The \fBpipe()\fR and \fBpipe2()\fR functions will fail if:
84.sp
85.ne 2
86.na
87 \fB0_BEMFILE\fR
88.ad
89.RS 10n
90 More than {\fB0_OPEN_MAX\fR} file descriptors are already in use by this process.
91.RE

93.sp
94.ne 2
95.na
96 \fB0_BENFILE\fR
97.ad
98.RS 10n
99 The number of simultaneously open files in the system would exceed a
100 system-imposed limit.
101.RE

103.sp
104.ne 2
105.na
106 \fB0_BEDEFAULT\fR
107.ad
108.RS 10n
109 The \fBifildes[2]\fR argument points to an illegal address.
110.RE

112.sp
113.LP
114 The \fBpipe2()\fR function will also fail if:
115.sp
116.ne 2
117.na
118 \fB0_BEINVAL\fR
119.ad
120.RS 10n
121 The \fBiflags\fR argument is illegal. Valid \fBiflags\fR are zero or a
```

```
122 bitwise inclusive-OR of \fBO_CLOEXEC\fR and \fBO_NONBLOCK\fR.
123 .RE

126 .SH ATTRIBUTES
127 .sp
128 .LP
129 See \fBattributes\fR(5) for descriptions of the following attributes:
130 .sp

132 .sp
133 .TS
134 box;
135 c | c
136 l | l .
137 ATTRIBUTE TYPE ATTRIBUTE VALUE
138 _
139 Interface Stability Standard
140 _
141 MT-Level Async-Signal-Safe
142 .TE

144 .SH SEE ALSO
145 .sp
146 .LP
147 \fBsh\fR(1), \fBfcntl\fR(2), \fBfstat\fR(2), \fBgetmsg\fR(2), \fBpoll\fR(2),
148 \fBputmsg\fR(2), \fBread\fR(2), \fBwrite\fR(2), \fBopen\fR(2),
149 \fBattributes\fR(5), \fBstandards\fR(5), \fBstreamio\fR(7I)
150 .SH NOTES
151 .sp
152 .LP
153 Since a pipe is bi-directional, there are two separate flows of data.
154 Therefore, the size (\fBst_size\fR) returned by a call to \fBfstat\fR(2) with
155 argument \fIfildes\fR[\fB0\fR] or \fIfildes\fR[\fB1\fR] is the number of bytes
156 available for reading from \fIfildes\fR[\fB0\fR] or \fIfildes\fR[\fB1\fR]
157 respectively. Previously, the size (\fBst_size\fR) returned by a call to
158 \fBfstat()\fR with argument \fIfildes\fR[\fB1\fR] (the write-end) was the
159 number of bytes available for reading from \fIfildes\fR[\fB0\fR] (the
160 read-end).
```

new/usr/src/man/man3c/btowc.3c

1

```
*****  
    3682 Sat Jul 19 19:24:36 2014  
new/usr/src/man/man3c/btowc.3c  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```

6351 Sat Jul 19 19:24:36 2014

new/usr/src/man/man3c/fgetwc.3c

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 \" te
2 .\" Copyright 2014 Garrett D'Amore <garrett@damore.org>
3 .\" Copyright (c) 2003, X/Open Company Limited. All Rights Reserved. Portions
4 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
5 .\" http://www.opengroup.org/bookstore/.
6 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
7 .\" This notice shall appear on any product containing this material.
8 .\" The contents of this file are subject to the terms of the Common Development
9 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
10 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
11 .TH FGETWC 3C "Jun 24, 2014"
12 .SH NAME
13 fgetwc, fgetwc_l \- get a wide-character code from a stream
14 .SH SYNOPSIS
15 .LP
16 .nf
17 #include <stdio.h>
18 #include <wchar.h>

20 \fBwint_t\fR \fBfgetwc\fR(\fBFILE *\fR\fIstream\fR);
21 .fi
22 .LP
23 .nf
24 #include <stdio.h>
25 #include <wchar.h>
26 #include <xlocale.h>

28 \fBwint_t\fR \fBfgetwc_l\fR(\fBFILE *\fR\fIstream\fR, \fBlocale_t\fR, \fIloc\fR)
29 .fi

31 .SH DESCRIPTION
32 .sp
33 .LP
34 The \fBfgetwc()\fR and \fBfgetwc_l()\fR functions obtain the next
35 character (if present) from the input stream pointed to by \fIstream\fR,
36 convert that to the corresponding wide-character code and advance the
37 associated file position indicator for the stream (if defined).
38 Whereas \fBfgetwc()\fR uses the current locale, \fBfgetwc_l()\fR uses the
39 locale specified by \fIloc\fR.
39 locale specified by \fIloc\fR.
40 .LP
41 If an error occurs, the resulting value of the file position indicator for the
42 stream is indeterminate.
43 .LP
44 The \fBfgetwc()\fR and \fBfgetwc_l()\fR functions may mark the \fBst_atime\fR
45 field of the file
46 associated with \fIstream\fR for update. The \fBst_atime\fR field will be
47 marked for update by the first successful execution of \fBfgetwc()\fR,
48 \fBfgetc\fR(3C), \fBfgetc_s\fR(3C), \fBfgetws\fR(3C), \fBfread\fR(3C),
49 \fBfscanf\fR(3C), \fBfgetc_r\fR(3C), \fBfgetchar_r\fR(3C), \fBfgetc_s\fR(3C), or
50 \fBscanf\fR(3C) using \fIstream\fR that returns data not supplied by a prior
51 call to \fBungetc\fR(3C) or \fBungetwc\fR(3C).
52 .SH RETURN VALUES
53 .LP
54 Upon successful completion both functions return the
55 wide-character code of the character read from the input stream pointed to by

```

```

56 \fIstream\fR converted to a type \fBwint_t\fR.
57 .LP
58 For standard-conforming (see \fBstandards\fR(5)) applications, if the
59 end-of-file indicator for the stream is set, \fBfgetwc()\fR and
60 \fBfgetwc_l()\fR return \fBWEOF\fR whether or not the stream is at
61 end-of-file.
62 .LP
63 If a read error occurs, the error indicator for the stream is set,
64 \fBfgetwc()\fR and \fBfgetwc_l()\fR returns \fBWEOF\fR and sets
65 \fBerrno\fR to indicate the error.
66 .LP
67 If an encoding error occurs, the error indicator for the stream is set,
68 \fBfgetwc()\fR and \fBfgetwc_l()\fR return \fBWEOF\fR, and \fBerrno\fR is
69 set to indicate the error.
70 .SH ERRORS
71 .LP
72 The \fBfgetwc()\fR and \fBfgetwc_l()\fR functions will fail if data needs to be
73 read and:
74 .TP
75 .B EAGAIN
76 The \fBONONBLOCK\fR flag is set for the file descriptor underlying
77 \fIstream\fR and the process would be delayed in the \fBfgetwc()\fR or
78 \fBfgetwc_l()\fR operation.
79 .TP
80 .B EBADF
81 The file descriptor underlying \fIstream\fR is not a valid file descriptor open
82 for reading.
83 .TP
84 .B EINTR
85 The read operation was terminated due to the receipt of a signal, and no data
86 was transferred.
87 .TP
88 .B EIO
89 A physical I/O error has occurred, or the process is in a background process
90 group attempting to read from its controlling terminal and either the process
91 is ignoring or blocking the \fBSIGTIN\fR signal or the process group is
92 orphaned.
93 .TP
94 .B EOVERFLOW
95 The file is a regular file and an attempt was made to read at or beyond the
96 offset maximum associated with the corresponding \fIstream\fR.
97 .LP
98 The \fBfgetwc()\fR and \fBfgetwc_l()\fR functions may fail if:
99 .TP
100 .B ENOMEM
101 Insufficient memory is available.
102 .TP
103 .B ENXIO
104 A request was made of a non-existent device, or the request was outside the
105 capabilities of the device.
106 .TP
107 .B EILSEQ
108 The data obtained from the input stream does not form a valid character.
109 .SH USAGE
110 .sp
111 .LP
112 The \fBferror\fR(3C) or \fBfeof\fR(3C) functions must be used to distinguish
113 between an error condition and an end-of-file condition.
114 .SH ATTRIBUTES
115 .sp
116 .LP
117 See \fBattributes\fR(5) for descriptions of the following attributes:
118 .TS
119 box;
120 c | c
121 1 | 1 .

```

```
122 ATTRIBUTE TYPE  ATTRIBUTE VALUE
123 _
124 CSI      Enabled
125 _
126 Interface Stability    See below.
127 _
128 MT-Level      MT-Safe
129 .TE

131 .LP
132 The
133 .B fgetwc()
134 function is Standard.  The
135 .B fgetwc_l()
136 function is Uncommitted.
137 .SH SEE ALSO
138 .LP
139 \fBfeof\fR(3C), \fBferror\fR(3C), \fBfgetc\fR(3C), \fBfgets\fR(3C),
140 \fBfgetws\fR(3C), \fBfopen\fR(3C), \fBfread\fR(3C), \fBfscanf\fR(3C),
141 \fBgetc\fR(3C), \fBgetchar\fR(3C), \fBgets\fR(3C), \fBscanf\fR(3C),
142 \fBnewlocale\fR(3C), \fBsetlocale\fR(3C), \fBungetc\fR(3C), \fBungetwc\fR(3C),
143 \fBuselocale\fR(3C), \fBattributes\fR(5),
144 \fBstandards\fR(5)
```

new/usr/src/man/man3c/fopen.3c

1

```
*****  
12307 Sat Jul 19 19:24:36 2014  
new/usr/src/man/man3c/fopen.3c  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```


new/usr/src/man/man3c/iswctype.3c

1

```
*****  
  4950 Sat Jul 19 19:24:36 2014  
new/usr/src/man/man3c/iswctype.3c  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```

new/usr/src/man/man3c/mbrtowc.3c

1

```
*****  
6226 Sat Jul 19 19:24:36 2014  
new/usr/src/man/man3c/mbrtowc.3c  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```

new/usr/src/man/man3c/mbsrtowcs.3c

1

```
*****  
6940 Sat Jul 19 19:24:36 2014  
new/usr/src/man/man3c/mbsrtowcs.3c  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```

new/usr/src/man/man3c/newlocale.3c

1

```
*****  
4309 Sat Jul 19 19:24:36 2014  
new/usr/src/man/man3c/newlocale.3c  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_
```

4621 Sat Jul 19 19:24:37 2014

new/usr/src/man/man3c/strcoll.3c

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1  \" te
2  .\" Copyright 2014 Garrett D'Amore <garrett@damore.org>
3  .\" Copyright (c) 2003, Sun Microsystems, Inc. All Rights Reserved.
4  .\" Copyright 1989 AT&T
5  .\" Portions Copyright (c) 2001, the Institute of Electrical and Electronics Eng
6  .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
7  .\" http://www.opengroup.org/bookstore/.
8  .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
9  .\" This notice shall appear on any product containing this material.
10 .\" The contents of this file are subject to the terms of the Common Development
11 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
12 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
13 .TH STRCOLL 3C \"Jun 23, 2014\"
14 .SH NAME
15 strcoll, strcoll_l \- string collation
16 .SH SYNOPSIS
17 .LP
18 .nf
19 #include <string.h>

21 \fBint\fR \fBstrcoll\fR(\fBconst char *\fR\fIs1\fR, \fBconst char *\fR\fIs2\fR);
22 .fi
23 .LP
24 .nf
25 \fBint\fR \fBstrcoll_l\fR(\fBconst char *\fR\fIs1\fR, \fBconst char *\fR\fIs2\fR
26 .fi

28 .SH DESCRIPTION
29 .LP
30 Both \fBstrcoll()\fR and \fBstrxfrm\fR(3C) provide for locale-specific string
31 sorting. \fBstrcoll()\fR is intended for applications in which the number of
32 comparisons per string is small. When strings are to be compared a number of
33 times, \fBstrxfrm\fR(3C) is a more appropriate function because the
34 transformation process occurs only once.
35 .LP
36 The \fBstrcoll_l()\fR function behaves
37 identically to \fBstrcoll()\fR, except instead of operating in the current
38 locale, it operates in the locale specified by \fIloc\fR.
39 .LP
40 The \fBstrcoll()\fR function does not change the setting of \fBerrno\fR if
41 successful.
42 .LP
43 Since no return value is reserved to indicate an error, an application wishing
44 to check for error situations should set \fBerrno\fR to 0, then call
45 \fBstrcoll()\fR, then check \fBerrno\fR.
46 .SH RETURN VALUES
47 .LP
48 Upon successful completion, \fBstrcoll()\fR returns an integer greater than,
49 equal to, or less than zero in direct correlation to whether string \fIs1\fR is
50 greater than, equal to, or less than the string \fIs2\fR. The comparison is
51 based on strings interpreted as appropriate to the locale
52 category \fBLC_COLLATE\fR (see \fBsetlocale\fR(3C)).
53 .LP
54 On error, \fBstrcoll()\fR may set \fBerrno\fR, but no return value is reserved
55 to indicate an error.
56 .SH ERRORS

```

```

57 .sp
58 .LP
59 The \fBstrcoll()\fR and \fBstrcoll_l()\fR functions may fail if:
60 .sp
61 .ne 2
62 .na
63 \fBEBINVAL\fR
64 .ad
65 .RS 10n
66 The \fIs1\fR or \fIs2\fR arguments contain characters outside the domain of the
67 collating sequence.
68 .RE
69 .SH FILES
70 .IP \fB/usr/lib/locale/\fR\fIlocale\fR/\fB/LC_COLLATE/*\fR
71 collation database for \fIlocale\fR
72 .RE

72 .SH ATTRIBUTES
73 .LP
74 See \fBattributes\fR(5) for descriptions of the following attributes:
75 .TS
76 box;
77 c | c
78 l | l .
79 ATTRIBUTE TYPE ATTRIBUTE VALUE
80 -
81 CSI Enabled
82 -
83 Interface Stability Standard
84 -
85 MT-Level MT-Safe
86 .TE

88 .SH SEE ALSO
89 .sp
90 .LP
91 \fBlocaledef\fR(1), \fBnewlocale\fR(3C), \fBsetlocale\fR(3C), \fBstring\fR(3C),
92 \fBstrxfrm\fR(3C), \fBuselocale\fR(3C),
93 \fBwsxfrm\fR(3C), \fBattributes\fR(5), \fBenviron\fR(5), \fBstandards\fR(5)

```

new/usr/src/man/man3c/string.3c

1

```
*****  
21322 Sat Jul 19 19:24:37 2014  
new/usr/src/man/man3c/string.3c  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```

```

*****
1767 Sat Jul 19 19:24:37 2014
new/usr/src/man/man3c/towlower.3c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 \" te
2 .\"
3 .\" This file and its contents are supplied under the terms of the
4 .\" Common Development and Distribution License (\"CDDL\"), version 1.0.
5 .\" You may only use this file in accordance with the terms of version
6 .\" 1.0 of the CDDL.
7 .\"
8 .\" A full copy of the text of the CDDL should have accompanied this
9 .\" source. A copy of the CDDL is also available via the Internet at
10 .\" http://www.illumos.org/license/CDDL.
11 .\"
12 .\"
13 .\" Copyright (c) 2014 Joyent, Inc. All rights reserved.
14 .\" Copyright 2014 Garrett D'Amore <garrett@damore.org>
15 .\"
16 .TH TOWLOWER 3C \"Jun 21, 2014\"
17 .SH NAME
18 tolower, tolower_l \- transliterate upper-case wide characters to lower-case
19 .SH SYNOPSIS
20 .LP
21 .nf
22 #include <wctype.h>

24 \fBwint_t\fR \fBtolower\fR(\fBwint_t\fR \fIwc\fR);
25 .fi
26 .LP
27 .nf
28 \fBwint_t\fR \fBtolower_l\fR(\fBwint_t\fR \fIwc\fR, \fBlocale_t\fR \fIloc\fR);
29 .fi
29 .nf
30 .SH DESCRIPTION
31 The function
32 .BR tolower()
33 is the wide character equivalent of the function
34 .BR tolower(3C).
35 It converts the upper-case wide character
36 .I wc
37 to the equivalent lower-case
38 wide character, if one exists. If one does not exist, it returns
39 .I wc
40 unchanged.
41 .LP
42 The function
43 .B tolower_l()
44 is equivalent to the function
45 .BR tolower() ,
46 but instead of operating in the current locale, operates in the
47 locale specified by
48 .IR loc .
49 .SH RETURN VALUES
50 On successful completion,
51 .B tolower()
52 and
53 .B tolower_l()
54 return the lower-case character that corresponds to the argument passed.
55 Otherwise, they return the argument unchanged.

```

```

56 .SH ERRORS
57 No errors are defined.
58 .SH ATTRIBUTES
59 .TS
60 box;
61 c | c
62 l | l .
63 ATTRIBUTE TYPE ATTRIBUTE VALUE
64 _
65 Interface Stability Standard
66 _
67 MT-Level MT-Safe
68 .TE

70 .SH SEE ALSO
71 .BR newlocale(3C),
72 .BR setlocale(3C),
73 .BR towupper(3C),
74 .BR uselocale(3C),
75 .BR locale(5)

```

new/usr/src/man/man3c/uselocale.3c

1

```
*****
2323 Sat Jul 19 19:24:37 2014
new/usr/src/man/man3c/uselocale.3c
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
_____unchanged_portion_omitted_____
```


new/usr/src/man/man3c/wcscoll.3c

1

4735 Sat Jul 19 19:24:37 2014

new/usr/src/man/man3c/wcscoll.3c

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

_____unchanged_portion_omitted_____

6901 Sat Jul 19 19:24:37 2014

new/usr/src/man/man3c/wcsrtombs.3c

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 \" te
2 .\" Copyright 2014 Garrett D'Amore <garrett@damore.org>
3 .\" Copyright (c) 1992, X/Open Company Limited. All Rights Reserved. Portions C
4 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
5 .\" http://www.opengroup.org/bookstore/.
6 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
7 .\" This notice shall appear on any product containing this material.
8 .\" The contents of this file are subject to the terms of the Common Development
9 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
10 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
11 .TH WCSRTOMBS 3C \"Jul 13, 2014\"
12 .SH NAME
13 wcsnrtombs, wcsnrtombs_l, wcsrtombs, wcsrtombs_l \- convert a wide-character str
14 (restartable)
15 .SH SYNOPSIS
16 .LP
17 .nf
18 #include <wchar.h>

20 \fBsize_t\fR \fBwcsrtombs\fR(\fBchar *restrict\fR \fFidst\fR, \fBconst wchar_t **
21 \fBsize_t\fR \fFilen\fR, \fBmbstate_t *restrict\fR \fFips\fR);
22 .fi
23 .LP
24 .nf
25 \fBsize_t\fR \fBwcsnrtombs_l\fR(\fBchar *restrict\fR \fFidst\fR, \fBconst wchar_t *
26 \fBsize_t\fR \fFinwc\fR, \fBsize_t\fR \fFilen\fR, \fBmbstate_t *restrict\fR \
27 .fi
28 .LP
29 .nf
30 #include <wchar.h>
31 #include <xlocale.h>

33 \fBsize_t\fR \fBwcsrtombs_l\fR(\fBchar *restrict\fR \fFidst\fR, \fBconst wchar_t
34 \fBsize_t\fR \fFilen\fR, \fBmbstate_t *restrict\fR \fFips\fR, \fBlocale_t\fR
35 .fi
36 .LP
37 .nf
38 \fBsize_t\fR \fBwcsnrtombs_l\fR(\fBchar *restrict\fR \fFidst\fR, \fBconst wchar_t
39 \fBsize_t\fR \fFinwc\fR, \fBsize_t\fR \fFilen\fR, \fBmbstate_t *restrict\fR \
40 .fi
41 .SH DESCRIPTION
42 .LP
43 The \fBwcsrtombs()\fR function converts a sequence of wide-characters from the
44 array indirectly pointed to by \fFisrc\fR into a sequence of corresponding
45 characters, beginning in the conversion state described by the object pointed
46 to by \fFips\fR. If \fFidst\fR is not a null pointer, the converted characters
47 are then stored into the array pointed to by \fFidst\fR. Conversion continues up
48 to and including a terminating null wide-character, which is also stored.
49 Conversion stops earlier in the following cases:
50 .RS +4
51 .TP
52 .ie t \(\bu
53 .el o
54 When a code is reached that does not correspond to a valid character.
55 .RE
56 .RS +4

```

```

57 .TP
58 .ie t \(\bu
59 .el o
60 When the next character would exceed the limit of \fFilen\fR total bytes to be
61 stored in the array pointed to by \fFidst\fR (and \fFidst\fR is not a null
62 pointer).
63 .RE
64 .RS +4
65 .TP
66 .ie t \(\bu
67 .el o
68 In the case of \fBwcsnrtombs()\fR and \fBwcsnrtombs_l()\fR, when \fFinwc\fR
69 wide characters have been completely converted.
70 .RE
71 .LP
72 Each conversion takes place as if by a call to the \fBwcbrtomb()\fR function.
73 .LP
74 If \fFidst\fR is not a null pointer, the pointer object pointed to by \fFisrc\fR
75 is assigned either a null pointer (if conversion stopped due to reaching a
76 terminating null wide-character) or the address just past the last
77 wide-character converted (if any). If conversion stopped due to reaching a
78 terminating null wide-character, the resulting state described is the initial
79 conversion state.
80 .LP
81 If \fFips\fR is a null pointer, these functions use their own
82 internal \fBmbstate_t\fR object, which is initialized at program startup to the
83 initial conversion state. Otherwise, the \fBmbstate_t\fR object pointed to by
84 \fFips\fR is used to completely describe the current conversion state of the
85 associated character sequence. The system will behave as if no function defined
86 in the Reference Manual calls any of these functions.
87 .LP
88 The behavior of \fBwcsrtombs()\fR and \fBwcsnrtombs()\fR are affected by the
89 \fBLC_CTYPE\fR category of the current locale. See \fBenviron\fR(5).
90 .LP
91 The \fBwcsrtombs_l()\fR and \fBwcsnrtombs_l()\fR functions behave identically
92 to \fBwcsrtombs()\fR and \fBwcsnrtombs()\fR respectively, except
93 that instead of operating in the current locale, they operate in the locale
94 specified by \fFiloc\fR.
95 .SH RETURN VALUES
96 .LP
97 If conversion stops because a code is reached that does not correspond to a
98 valid character, an encoding error occurs. In this case, these
99 functions store the value of the macro \fBEILSEQ\fR in \fBerrno\fR and return
100 \fB(size_t)\fR(\fBmfil\fR); the conversion state is undefined. Otherwise, they return
101 the number of bytes in the resulting character sequence, not including the
102 terminating null (if any).
103 .SH ERRORS
104 .LP
105 These functions may fail if:
106 .sp
107 .ne 2
108 .na
109 \fB\FBEINVAL\fR
110 .ad
111 .RS 10n
112 The \fFips\fR argument points to an object that contains an invalid conversion
113 state.
114 .RE

116 .sp
117 .ne 2
118 .na
119 \fB\FBEILSEQ\fR
120 .ad
121 .RS 10n
122 A wide-character code does not correspond to a valid character.

```

```
123 .RE
124 .SH ATTRIBUTES
125 .LP
126 See \fBattributes\fR(5) for descriptions of the following attributes:
127 .TS
128 box;
129 c | c
130 l | l .
131 ATTRIBUTE TYPE ATTRIBUTE VALUE
132 _
133 Interface Stability See below.
134 _
135 MT-Level See below.
136 .TE

138 .LP
139 The \fBwcsrtombs()\fR and \fBwcsnrtombs()\fR functions are Standard. The
140 \fBwcsrtombs_l()\fR and \fBwcsnrtombs_l()\fR functions are Uncommitted.
141 .LP
142 If \fIps\fR is a null pointer, these functions should be considered Unsafe
143 for use in multithreaded applications. Otherwise, they are MT-Safe.
144 .SH SEE ALSO
145 .LP
146 \fBmbsinit\fR(3C), \fBnewlocale\fR(3C), \fBsetlocale\fR(3C), \fBuselocale\fR(3C)
147 \fBwcrctomb\fR(3C), \fBattributes\fR(5),
148 \fBenviron\fR(5), \fBstandards\fR(5)
```

new/usr/src/man/man3lib/libc.3lib

1

```
*****  
34015 Sat Jul 19 19:24:37 2014  
new/usr/src/man/man3lib/libc.3lib  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```

2499 Sat Jul 19 19:24:38 2014

new/usr/src/man/man4/init.4

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1  \" te
2  .\" Copyright 2014 Garrett D'Amore
3  .\" Copyright (c) 2003, Sun Microsystems, Inc. All Rights Reserved.
4  .\" Copyright 1989 AT&T
5  .\" The contents of this file are subject to the terms of the Common Development
6  .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
7  .\" When distributing Covered Code, include this CDDL HEADER in each file and in
8  .TH INIT 4 "Mar 15, 2014"
8  .TH init 4 "Mar 15, 2014"
9  .SH NAME
10 init \- set default system time zone and locale
11 .SH SYNOPSIS
12 .LP
13 .nf
14 \fB/etc/default/init\fR
15 .fi

17 .SH DESCRIPTION
18 .sp
19 .LP
20 This file sets the time zone environment variable \fBTZ\fR, and the
21 locale-related environment variables \fBLANG\fR, \fBLC_COLLATE\fR,
22 \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, \fBLC_MONETARY\fR, \fBLC_NUMERIC\fR, and
23 \fBLC_TIME\fR.
24 .sp
25 .LP
26 \fB/etc/TIMEZONE\fR is a symbolic link to \fB/etc/default/init\fR. This
27 link exists for compatibility with legacy software, is obsolete, and may
28 be removed in a future release.
29 .sp
30 .LP
31 The number of environment variables that can be set from
32 \fB/etc/default/init\fR is limited to 20.
33 .sp
34 .LP
35 The format of the file is:
36 .sp
37 .in +2
38 .nf
39 \fIVAR\fR\fB=\fR\fIvalue\fR
40 .fi
41 .in -2
42 .sp

44 .sp
45 .LP
46 where \fIVAR\fR is a timezone environment variable and \fIvalue\fR is the value
47 assigned to the variable. \fIvalue\fR can be enclosed in double quotes (") or
48 single quotes (&'). The double or single quotes cannot be part of the value.
49 .SH SEE ALSO
50 .sp
51 .LP
52 \fBinit\fR(1M), \fBrtc\fR(1M), \fBctime\fR(3C), \fBenviron\fR(5)
53 .SH NOTES
54 .sp
55 .LP

```

```

56 When changing the \fBTZ\fR setting on x86 systems, you must make a
57 corresponding change to the \fB/etc/rtc_config\fR file to account for the new
58 timezone setting. This can be accomplished by executing the following commands,
59 followed by a reboot, to make the changes take effect:
60 .sp
61 .in +2
62 .nf
63 # rtc \fB-z\fR \fIzone-name\fR
64 # rtc \fB-c\fR

66 .fi
67 .in -2
68 .sp

70 .sp
71 .LP
72 where \fIzone-name\fR is the same name as the \fBTZ\fR variable setting.
73 .sp
74 .LP
75 See \fBrtc\fR(1M) for information on the \fBrtc\fR command.

```

```

*****
4030 Sat Jul 19 19:24:38 2014
new/usr/src/man/man5/Makefile
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****

```

```

1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet
9 # at http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2011, Richard Lowe
14 # Copyright (c) 2012 by Delphix. All rights reserved.
15 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
16 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
17 #
18 #
19 include      $(SRC)/Makefile.master
20 #
21 MANSECT=     5
22 #
23 MANFILES=    Intro.5           //
24              acl.5             //
25              ad.5              //
26              ascii.5          //
27              attributes.5     //
28              audit_binfile.5  //
29              audit_remote.5   //
30              audit_syslog.5   //
31              brands.5         //
32              cancellation.5   //
33              charmap.5        //
34              condition.5      //
35              crypt_bsdbf.5    //
36              crypt_bsdmd5.5   //
37              crypt_sha256.5   //
38              crypt_sha512.5   //
39              crypt_sunmd5.5   //
40              crypt_unix.5     //
41              device_clean.5   //
42              dhcp.5           //
43              dhcp_modules.5   //
44              environ.5       //
45              eqn.5            //
46              eqnchar.5        //
47              extendedFILE.5   //
48              filesystem.5     //
49              fnmatch.5        //
50              formats.5        //
51              fsattr.5         //
52              grub.5           //
53              gss_auth_rules.5 //
54              hal.5            //
55              iconv.5          //
56              iconv_unicode.5  //

```

```

57              ieee802.11.5     //
58              ipfilter.5       //
59              isalist.5        //
60              kerberos.5       //
61              krb5_auth_rules.5 //
62              krb5envvar.5     //
63              largefile.5      //
64              lf64.5           //
65              lfcompile.5       //
66              lfcompile64.5     //
67              locale.5         //
68              man.5            //
69              mandoc_char.5     //
70              mandoc_roff.5     //
71              mdoc.5           //
72              mansun.5         //
73              me.5             //
74              mech_spnego.5    //
75              mm.5             //
76              ms.5             //
77              mutex.5          //
78              nfssec.5         //
79              pam_allow.5       //
80              pam_authok_check.5 //
81              pam_authok_get.5  //
82              pam_authok_store.5 //
83              pam_deny.5       //
84              pam_dhkeys.5     //
85              pam_dial_auth.5   //
86              pam_krb5.5        //
87              pam_krb5_migrate.5 //
88              pam_ldap.5       //
89              pam_list.5       //
90              pam_passwd_auth.5 //
91              pam_rhosts_auth.5 //
92              pam_roles.5      //
93              pam_sample.5     //
94              pam_smb_passwd.5  //
95              pam_smbfs_login.5 //
96              pam_tsol_account.5 //
97              pam_unix_account.5 //
98              pam_unix_auth.5   //
99              pam_unix_cred.5   //
100             pam_unix_session.5 //
101             pkcs11_kernel.5    //
102             pkcs11_softtoken.5 //
103             privileges.5      //
104             prof.5            //
105             rbac.5            //
106             regex.5           //
107             regexp.5          //
108             resource_controls.5 //
109             smf.5             //
110             smf_bootstrap.5    //
111             smf_method.5       //
112             smf_restarter.5    //
113             smf_security.5     //
114             smf_template.5     //
115             standards.5       //
116             sticky.5          //
117             tbl.5             //
118             tecla.5           //
119             term.5            //
120             threads.5         //
121             trusted_extensions.5 //

```

new/usr/src/man/man5/Makefile

3

```

122          vgrindefs.5          \
123          zones.5              \
124          zpool-features.5

126 MANLINKS=  ANSI.5           \
127              C++.5           \
128              C.5             \
129              CSI.5           \
130              ISO.5           \
131              MT-Level.5     \
132              POSIX.1.5     \
133              POSIX.2.5     \
134              POSIX.5       \
135              RBAC.5        \
136              SUS.5         \
137              SUSv2.5       \
138              SUSv3.5       \
139              SVID.5        \
140              SVID3.5       \
141              XNS.5         \
142              XNS4.5        \
143              XNS5.5        \
144              XPG.5         \
145              XPG3.5        \
146              XPG4.5        \
147              XPG4v2.5     \
148              advance.5     \
149              architecture.5 \
150              availability.5 \
151              compile.5     \
152              intro.5       \
153              pthreads.5    \
154              stability.5   \
155              standard.5    \
156              step.5        \
157              teclarc.5

159 intro.5      := LINKSRC = Intro.5

161 CSI.5        := LINKSRC = attributes.5
162 MT-Level.5   := LINKSRC = attributes.5
163 architecture.5 := LINKSRC = attributes.5
164 availability.5 := LINKSRC = attributes.5
165 stability.5   := LINKSRC = attributes.5
166 standard.5   := LINKSRC = attributes.5

168 RBAC.5       := LINKSRC = rbac.5

170 advance.5    := LINKSRC = regexp.5
171 compile.5    := LINKSRC = regexp.5
172 step.5       := LINKSRC = regexp.5

174 ANSI.5       := LINKSRC = standards.5
175 C++.5        := LINKSRC = standards.5
176 C.5          := LINKSRC = standards.5
177 ISO.5        := LINKSRC = standards.5
178 POSIX.1.5    := LINKSRC = standards.5
179 POSIX.2.5    := LINKSRC = standards.5
180 POSIX.5      := LINKSRC = standards.5
181 SUS.5        := LINKSRC = standards.5
182 SUSv2.5      := LINKSRC = standards.5
183 SUSv3.5      := LINKSRC = standards.5
184 SVID.5       := LINKSRC = standards.5
185 SVID3.5      := LINKSRC = standards.5
186 XNS.5        := LINKSRC = standards.5
187 XNS4.5       := LINKSRC = standards.5

```

new/usr/src/man/man5/Makefile

4

```

188 XNS5.5      := LINKSRC = standards.5
189 XPG.5       := LINKSRC = standards.5
190 XPG3.5      := LINKSRC = standards.5
191 XPG4.5      := LINKSRC = standards.5
192 XPG4v2.5   := LINKSRC = standards.5

194 teclarc.5  := LINKSRC = tecla.5

196 pthreads.5 := LINKSRC = threads.5

198 .KEEP_STATE:

200 include     $(SRC)/man/Makefile.man

202 install:    $(ROOTMANFILES) $(ROOTMANLINKS)

```

7225 Sat Jul 19 19:24:38 2014

new/usr/src/man/man5/eqn.5

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 .\"
2 .\" Permission to use, copy, modify, and distribute this software for any
3 .\" purpose with or without fee is hereby granted, provided that the above
4 .\" copyright notice and this permission notice appear in all copies.
5 .\"
6 .\" THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
7 .\" WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
8 .\" MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
9 .\" ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
10 .\" WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
11 .\" ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
12 .\" OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
13 .\"
14 .\"
15 .\" Copyright (c) 2011 Kristaps Dzonsons <kristaps@bsd.lv>
16 .\" Copyright 2012 Nexenta Systems, Inc. All rights reserved.
17 .\"
18 .Dd Jul 19, 2014
19 .Dt EQN 5
20 .Os
21 .Sh NAME
22 .Nm eqn
23 .Nd eqn language reference for mandoc
24 .Sh DESCRIPTION
25 The
26 .Nm eqn
27 language is an equation-formatting language.
28 It is used within
29 .Xr mdoc 5
30 and
31 .Xr man 5
32 .Ux
33 manual pages.
34 It describes the
35 .Em structure
36 of an equation, not its mathematical meaning.
37 This manual describes the
38 .Nm
39 language accepted by the
40 .Xr mandoc 1
41 utility, which corresponds to the Second Edition eqn specification (see
42 .Sx SEE ALSO
43 for references).
44 .Pp
45 Equations within
46 .Xr mdoc 5
47 or
48 .Xr man 5
49 documents are enclosed by the standalone
50 .Sq \&.EQ
51 and
52 .Sq \&.EN
53 tags.
54 Equations are multi-line blocks consisting of formulas and control
55 statements.
56 .Sh EQUATION STRUCTURE

```

```

57 Each equation is bracketed by
58 .Sq \&.EQ
59 and
60 .Sq \&.EN
61 strings.
62 .Em Note :
63 these are not the same as
64 .Xr roff 5
65 macros, and may only be invoked as
66 .Sq \&.EQ .
67 .Pp
68 The equation grammar is as follows, where quoted strings are
69 case-sensitive literals in the input:
70 .Bd -literal -offset indent
71 eqn      : box | eqn box
72 box      : text
73           | \*q{\*q eqn \*q}\*q
74           | \*qdefine\*q text text
75           | \*qndefine\*q text text
76           | \*qtdefine\*q text text
77           | \*qgfont\*q text
78           | \*qgsize\*q text
79           | \*qset\*q text text
80           | \*qundef\*q text
81           | box pos box
82           | box mark
83           | \*qmatrix\*q \*q{\*q [col \*q{\*q list \*q}\*q] \*q *}
84           | pile \*q{\*q list \*q}\*q
85           | font box
86           | \*qsize\*q text box
87           | \*qlleft\*q text eqn [\*qright\*q text]
88 col      : \*qlcol\*q | \*qrcol\*q | \*qccol\*q | \*qcol\*q
89 text     : [\*space\*q]+ | \*q\*q.\*q\*q
90 pile     : \*qlpile\*q | \*qcpile\*q | \*qrpile\*q | \*qpile\*q
91 pos      : \*qover\*q | \*qsup\*q | \*qsub\*q | \*qto\*q | \*qfrom\*q
92 mark     : \*qdot\*q | \*qdotdot\*q | \*qhat\*q | \*qtilde\*q | \*qvec\*q
93           | \*qdyad\*q | \*qbar\*q | \*qunder\*q
94 font     : \*qroman\*q | \*qitalic\*q | \*qbold\*q | \*qfat\*q
95 list     : eqn
96           | list \*qabove\*q eqn
97 space    : [\*e^~ \*et]
98 .Ed
99 .Pp
100 White-space consists of the space, tab, circumflex, and tilde
101 characters.
102 If within a quoted string, these space characters are retained.
103 Quoted strings are also not scanned for replacement definitions.
104 .Pp
105 The following text terms are translated into a rendered glyph, if
106 available: alpha, beta, chi, delta, epsilon, eta, gamma, iota, kappa,
107 lambda, mu, nu, omega, omicron, phi, pi, psi, rho, sigma, tau, theta,
108 upsilon, xi, zeta, DELTA, GAMMA, LAMBDA, OMEGA, PHI, PI, PSI, SIGMA,
109 THETA, UPSILON, XI, inter (intersection), union (union), prod (product),
110 int (integral), sum (summation), grad (gradient), del (vector
111 differential), times (multiply), cdot (centre-dot), nothing (zero-width
112 space), approx (approximately equals), prime (prime), half (one-half),
113 partial (partial differential), inf (infinity), >> (much greater), <<
114 (much less), \-> (left arrow), <-\ (right arrow), += (plus-minus), !=
115 (not equal), == (equivalence), <= (less-than-equal), and >=
116 (more-than-equal).
117 .Pp
118 The following control statements are available:
119 .Bl -tag -width Ds
120 .It Cm define
121 Replace all occurrences of a key with a value.
122 Its syntax is as follows:

```



```

123 .Pp
124 .Dl define Ar key cvalc
125 .Pp
126 The first character of the value string,
127 .Ar c ,
128 is used as the delimiter for the value
129 .Ar val .
130 This allows for arbitrary enclosure of terms (not just quotes), such as
131 .Pp
132 .Dl define Ar foo 'bar baz'
133 .Dl define Ar foo cbar bazc
134 .Pp
135 It is an error to have an empty
136 .Ar key
137 or
138 .Ar val .
139 Note that a quoted
140 .Ar key
141 causes errors in some
142 .Nm
143 implementations and should not be considered portable.
144 It is not expanded for replacements.
145 Definitions may refer to other definitions; these are evaluated
146 recursively when text replacement occurs and not when the definition is
147 created.
148 .Pp
149 Definitions can create arbitrary strings, for example, the following is
150 a legal construction.
151 .Bd -literal -offset indent
152 define foo 'define'
153 foo bar 'baz'
154 .Ed
155 .Pp
156 Self-referencing definitions will raise an error.
157 The
158 .Cm ndefine
159 statement is a synonym for
160 .Cm define ,
161 while
162 .Cm tdefine
163 is discarded.
164 .It Cm gfont
165 Set the default font of subsequent output.
166 Its syntax is as follows:
167 .Pp
168 .Dl gfont Ar font
169 .Pp
170 In
171 .Xr mandoc 1 ,
172 this value is discarded.
173 .It Cm gsize
174 Set the default size of subsequent output.
175 Its syntax is as follows:
176 .Pp
177 .Dl gsize Ar size
178 .Pp
179 The
180 .Ar size
181 value should be an integer.
182 .It Cm set
183 Set an equation mode.
184 In
185 .Xr mandoc 1 ,
186 both arguments are thrown away.
187 Its syntax is as follows:
188 .Pp

```

```

189 .Dl set Ar key val
190 .Pp
191 The
192 .Ar key
193 and
194 .Ar val
195 are not expanded for replacements.
196 This statement is a GNU extension.
197 .It Cm undef
198 Unset a previously-defined key.
199 Its syntax is as follows:
200 .Pp
201 .Dl define Ar key
202 .Pp
203 Once invoked, the definition for
204 .Ar key
205 is discarded.
206 The
207 .Ar key
208 is not expanded for replacements.
209 This statement is a GNU extension.
210 .El
211 .Sh COMPATIBILITY
212 This section documents the compatibility of
213 .Xr mandoc 1
214 .Nm
215 and the
216 .Xr troff 1
217 .Nm
218 implementation (including GNU troff).
219 .Pp
220 .Bl -dash -compact
221 .It
222 The text string
223 .Sq \e*q
224 is interpreted as a literal quote in
225 .Xr troff 1 .
226 In
227 .Xr mandoc 1 ,
228 this is interpreted as a comment.
229 .It
230 In
231 .Xr troff 1 ,
232 The circumflex and tilde white-space symbols map to
233 fixed-width spaces.
234 In
235 .Xr mandoc 1 ,
236 these characters are synonyms for the space character.
237 .It
238 The
239 .Xr troff 1 ,
240 implementation of
241 .Nm
242 allows for equation alignment with the
243 .Cm mark
244 and
245 .Cm lineup
246 tokens.
247 .Xr mandoc 1
248 discards these tokens.
249 The
250 .Cm back Ar n ,
251 .Cm fwd Ar n ,
252 .Cm up Ar n ,
253 and
254 .Cm down Ar n

```

```
255 commands are also ignored.
256 .El
257 .Sh SEE ALSO
258 .Xr mandoc 1 ,
259 .Xr man 5 ,
260 .Xr mandoc_char 5 ,
261 .Xr mdoc 5 ,
262 .Xr roff 5
263 .Rs
264 .%A Brian W. Kernighan
265 .%A Lorinda L. Cherry
266 .%T System for Typesetting Mathematics
267 .%J Communications of the ACM
268 .%V 18
269 .%P 151\(\enl57
270 .%D March, 1975
271 .Re
272 .Rs
273 .%A Brian W. Kernighan
274 .%A Lorinda L. Cherry
275 .%T Typesetting Mathematics, User's Guide
276 .%D 1976
277 .Re
278 .Rs
279 .%A Brian W. Kernighan
280 .%A Lorinda L. Cherry
281 .%T Typesetting Mathematics, User's Guide (Second Edition)
282 .%D 1978
283 .Re
284 .Sh HISTORY
285 The eqn utility, a preprocessor for troff, was originally written by
286 Brian W. Kernighan and Lorinda L. Cherry in 1975.
287 The GNU reimplementation of eqn, part of the GNU troff package, was
288 released in 1989 by James Clark.
289 The eqn component of
290 .Xr mandoc 1
291 was added in 2011.
292 .Sh AUTHORS
293 This
294 .Nm
295 reference was written by
296 .An Kristaps Dzonsons ,
297 .Mt kristaps@bsd.lv .
```

8753 Sat Jul 19 19:24:38 2014

new/usr/src/man/man5/man.5

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 .\" Copyright 2014 Garrett D'Amore <garrett@damore.org>
1 '\\" te
2 .\" Copyright (c) 1995, Sun Microsystems, Inc.
3 .\" The contents of this file are subject to the terms of the Common Development
4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
6 .Dd \"Jul 19, 2014\"
7 .Dt MAN 5
8 .Os
9 .Sh NAME
10 .Nm man
11 .Nd macros to format Reference Manual pages
12 .Sh SYNOPSIS
13 .Nm mandoc
14 .Fl T Ar man
15 .Ar
16 .Nm nroff
17 .Fl man
18 .Ar
19 .Nm troff
20 .Fl man
21 .Ar
22 .Sh DESCRIPTION
6 .TH MAN 5 \"Jan 30, 1995\"
7 .SH NAME
8 man \- macros to format Reference Manual pages
9 .SH SYNOPSIS
10 .LP
11 .nf
12 \fBnroff\fR \fB-man\fR \fIfilename\fR...
13 .fi

15 .LP
16 .nf
17 \fBtroff\fR \fB-man\fR \fIfilename\fR...
18 .fi

20 .SH DESCRIPTION
21 .sp
22 .LP
23 These macros are used to lay out the reference pages in this manual. Note: if
24 .Ar file
25 contains format input for a preprocessor, the commands shown
24 \fIfilename\fR contains format input for a preprocessor, the commands shown
26 above must be piped through the appropriate preprocessor. This is handled
27 automatically by the
28 .Xr man 1
29 command. See the
30 .Sx Conventions
31 section.
32 .LP
33 Any text argument
34 .Ar t
35 may be zero to six words. Quotes may be used to
36 include SPACE characters in a
37 .Qq word .

```

```

38 If
39 .Ar text
40 is empty, the special
26 automatically by the \fBman\fR(1) command. See the ``Conventions`` section.
27 .sp
28 .LP
29 Any text argument \fIt\fR may be zero to six words. Quotes may be used to
30 include SPACE characters in a "word". If \fIttext\fR is empty, the special
41 treatment is applied to the next input line with text to be printed. In this
42 way
43 .Nm \&.I
44 may be used to italicize a whole line, or
45 .Nm \&.SB
46 may be used to make small bold letters.
47 .LP
32 way \fB&.I\fR may be used to italicize a whole line, or \fB&.SB\fR may be
33 used to make small bold letters.
34 .sp
35 .LP
48 A prevailing indent distance is remembered between successive indented
49 paragraphs, and is reset to default value upon reaching a non-indented
50 paragraph. Default units for indents
51 .Nm i
52 are ens.
53 .LP
38 paragraph. Default units for indents \fIi\fR are ens.
39 .sp
40 .LP
54 Type font and size are reset to default values before each paragraph, and after
55 processing font and size setting macros.
56 .Pp
57 These strings are predefined by
58 .Nm -man :
59 .Bl -tag -width Ds
60 .It Nm \e*R
61 .Sq \(\rg ,
62 .Sq (Reg)
63 in
64 .Nm nroff .
65 .It Nm \e*S
43 .sp
44 .LP
45 These strings are predefined by \fB-man\fR:
46 .sp
47 .ne 2
48 .na
49 \fB\fB\e*R\fR\fR
50 .ad
51 .RS 8n
52 '\(\rg', '(Reg)' in \fBnroff\fR.
53 .RE

55 .sp
56 .ne 2
57 .na
58 \fB\fB\e*S\fR\fR
59 .ad
60 .RS 8n
66 Change to default type size.
67 .El
68 .Sh "Requests"
69 .RE

64 .SS "Requests"
65 .sp
66 .LP

```

```

69 * n.t.l. = next text line; p.i. = prevailing indent
70 .Bl -column ".TH n s d f m" "Cause " "t=n.t.l.*" "Explanation " -offset Ds
71 .It Sy Request Sy Cause Sy "If No" Sy Explanation
72 .It "" Sy Break Sy "Argument" ""
73 .It Nm \&.B Ar "t" no Ar t Ns =n.t.l.* Text is in bold font.
74 .It Nm \&.BI Ar t no Ar t Ns =n.t.l. Join words, alternating bold and
75 .It Nm \&.BR Ar t no Ar t Ns =n.t.l. Join words, alternating bold and
76 .It Nm \&.DT no Li \&.5i li... Restore default tabs.
77 .It Nm \&.HP Ar i yes Ar i Ns =p.i.* "Begin paragraph with hanging in
78 .It Nm \&.I Ar t no Ar t Ns =n.t.l. Text is italic.
79 .It Nm \&.IB Ar t no Ar t Ns =n.t.l. Join words, alternating italic a
80 .It Nm \&.IP Ar x Ar i yes Ar x Ns ="" Same as
81 .Nm \&.TP
82 with tag
83 .Ar x .
84 .It Nm \&.IR Ar t no Ar t Ns =n.t.l. Join words, alternating italic a
85 .It Nm \&.IX Ar t no - Index macro, not used (obsolete).
86 .It Nm \&.LP yes - Begin left-aligned paragraph. Set prevailing ind
87 .It Nm \&.P yes - Same as
88 .Nm \&.LP .
89 .It Nm \&.PD Ar d no Ar d Ns =.4v Set vertical distance between pa
90 .It Nm \&.PP yes - Same as
91 .Nm \&.LP .
92 .It Nm \&.RE yes - End of relative indent. Restores prevailing inde
93 .It Nm \&.RB Ar t no Ar t Ns =n.t.l. Join words, alternating roman an
94 .It Nm \&.RI Ar t no Ar t Ns =n.t.l. Join words, alternating roman an
95 .It Nm \&.RS Ar i yes Ar i Ns =p.i. Start relative indent, increase
96 Sets prevailing indent to .5i for nested indents.
97 .It Nm \&.SB Ar t no - Reduce size of text by 1 point, make tex
98 .It Nm \&.SH Ar t yes - Section Heading.
99 .It Nm \&.SM Ar t no Ar t Ns =n.t.l. Reduce size of text by 1 point.
100 .It Nm \&.SS Ar t yes Ar t Ns =n.t.l. Section Subheading.
101 .It Nm \&.TH Ar n s d f m yes - Begin reference page Ar n , No o
102 .It Nm \&.TP Ar i yes Ar i Ns =p.i. Begin indented paragraph, with t
103 .Ar i .
104 .It Nm \&.TX Ar t p no - Resolve the title abbreviation Ar t ; No
105 .El
106 .Ss "Conventions"
107 When formatting a manual page,
108 .Nm
109 examines the first line to determine
68 .sp

70 .sp
71 .TS
72 c c c c
73 c c c c .
74 \fIRequest\fR \fICause\fR \fIIf no\fR \fIExplanation\fR
75 \fIBreak\fR \fIArgument\fR
76 \fB\&.B \fR\fIt\fR no \fIt\fR=n.t.l.* Text is in bold font.
77 \fB\&.BI \fR\fIt\fR no \fIt\fR=n.t.l. Join words, alternating bold and
78 \fB\&.BR \fR\fIt\fR no \fIt\fR=n.t.l. Join words, alternating bold and
79 \fB\&.DT\fR no \&.5i li... Restore default tabs.
80 \fB\&.HP \fR\fIi\fR yes \fIi\fR=p.i.* T{
81 Begin paragraph with hanging indent. Set prevailing indent to \fIi\fR.
82 T}
83 \fB\&.I \fR\fIt\fR no \fIt\fR=n.t.l. Text is italic.
84 \fB\&.IB \fR\fIt\fR no \fIt\fR=n.t.l. Join words, alternating italic a
85 \fB\&.IP \fR\fIx i\fR yes \fIx\fR="" Same as \fB\&.TP\fR with tag \fI
86 \fB\&.IR \fR\fIt\fR no \fIt\fR=n.t.l. T{
87 Join words, alternating italic and roman.
88 T}
89 \fB\&.IX \fR\fIt\fR no - Index macro, for SunSoft internal use.
90 \fB\&.LP\fR yes - T{
91 Begin left-aligned paragraph. Set prevailing indent to .5i.
92 T}

```

```

93 \fB\&.P\fR yes - Same as \fB\&.LP\fR.
94 \fB\&.PD \fR\fId\fR no \fId\fR=.4v T{
95 Set vertical distance between paragraphs.
96 T}
97 \fB\&.PP\fR yes - Same as \fB\&.LP\fR.
98 \fB\&.RE\fR yes - T{
99 End of relative indent. Restores prevailing indent.
100 T}
101 \fB\&.RB \fR\fIt\fR no \fIt\fR=n.t.l. Join words, alternating roman an
102 \fB\&.RI \fR\fIt\fR no \fIt\fR=n.t.l. T{
103 Join words, alternating roman and italic.
104 T}
105 \fB\&.RS \fR\fIi\fR yes \fIi\fR=p.i. T{
106 Start relative indent, increase indent by \fIi\fR. Sets prevailing indent to .5i
107 T}
108 \fB\&.SB \fR\fIt\fR no - T{
109 Reduce size of text by 1 point, make text bold.
110 T}
111 \fB\&.SH \fR\fIt\fR yes - Section Heading.
112 \fB\&.SM \fR\fIt\fR no \fIt\fR=n.t.l. Reduce size of text by 1 point.
113 \fB\&.SS \fR\fIt\fR yes \fIt\fR=n.t.l. Section Subheading.
114 \fB\&.TH \fR\FIN S "f d, m\fR"
115 \fB\&.TH \fR\FIN s d f m\fR yes - T{
116 Begin reference page \fIn\fR, of of section \fIs\fR; \fId\fR is the date of the
117 T}
118 \fB\&.TP \fR\fIi\fR yes \fIi\fR=p.i. T{
119 Begin indented paragraph, with the tag given on the next text line. Set prevaili
120 T}
121 \fB\&.TX \fR\fIt \fR\fIp\fR no - T{
122 Resolve the title abbreviation \fIt\fR; join to punctuation mark (or text) \fIp\
123 T}
124 .TE

126 .Ss "Conventions"
127 .sp
128 .LP
129 When formatting a manual page, \fBman\fR examines the first line to determine
130 whether it requires special processing. For example a first line consisting of:
131 .Lp
132 .Dl \&'e" t
133 .Lp
134 indicates that the manual page must be run through the
135 .Xr tbl 1
136 .sp
137 .LP
138 \fB\&'e" t\fR
139 .sp
140 .LP
141 indicates that the manual page must be run through the \fBtbl\fR(1)
142 preprocessor.
143 .Lp
144 .sp
145 A typical manual page for a command or function is laid out as follows:
146 .Bl -tag -width ".SH RETURN VALUES"
147 .
148 .It Nm \&.TH Ar title Op "1-9"
149 .
150 .sp
151 .ne 2
152 .na
153 \fB\&.TH \fI TITLE \fR[1-9]\fR " , "
154 .ad
155 .RS 23n
156 The name of the command or function, which serves as the title of the manual
157 page. This is followed by the number of the section in which it appears.

```

```

125 .
126 .It Nm SH NAME
127 .
149 .RE

151 .sp
152 .ne 2
153 .na
154 \fB\&.SH NAME\fR
155 .ad
156 .RS 23n
128 The name, or list of names, by which the command is called, followed by a dash
129 and then a one-line summary of the action performed. All in roman font, this
130 section contains no
131 .Xr troff 1
132 commands or escapes, and no macro requests.
133 It is used to generate the database used by the
134 .Xr whatis 1
135 command.
136 .
137 .It Nm SH SYNOPSIS
138 .Bl -tag -width "Functions:"
139 .It Sy Commands:
159 section contains no \fBtroff\fR(1) commands or escapes, and no macro requests.
160 It is used to generate the \fBwindex\fR database, which is used by the
161 \fBwhatis\fR(1) command.
162 .RE

164 .sp
165 .ne 2
166 .na
167 \fB\&.SH SYNOPSIS\fR
168 .ad
169 .RS 23n
170 .sp
171 .ne 2
172 .na
173 \fBCommands:\fR
174 .ad
175 .RS 13n
140 The syntax of the command and its arguments, as typed on the command line.
141 When in boldface, a word must be typed exactly as printed. When in italics, a
142 word can be replaced with an argument that you supply. References to bold or
143 italicized items are not capitalized in other sections, even when they begin a
144 sentence.
145 .Lp
181 .sp
146 Syntactic symbols appear in roman face:
147 .Bl -tag -width " "
148 .It Op " "
183 .sp
184 .ne 2
185 .na
186 \fB[ ]\fR
187 .ad
188 .RS 13n
149 An argument, when surrounded by brackets is optional.
150 .It |
190 .RE

192 .sp
193 .ne 2
194 .na
195 \fB|\fR
196 .ad
197 .RS 13n

```

```

151 Arguments separated by a vertical bar are exclusive. You can supply only one
152 item from such a list.
153 .It \&.\|.\|.
200 .RE

202 .sp
203 .ne 2
204 .na
205 \fB\&.\|.\|.\fR
206 .ad
207 .RS 13n
154 Arguments followed by an ellipsis can be repeated. When an ellipsis follows a
155 bracketed set, the expression within the brackets can be repeated.
156 .El
157 .It Sy Functions:
158 If required, the data declaration, or
159 .Li #include
160 directive, is shown first,
210 .RE

212 .RE

214 .sp
215 .ne 2
216 .na
217 \fBFunctions:\fR
218 .ad
219 .RS 14n
220 If required, the data declaration, or \fB#include\fR directive, is shown first,
161 followed by the function declaration. Otherwise, the function declaration is
162 shown.
163 .El
164 .
165 .It Nm \&.SH DESCRIPTION
166 .
223 .RE

225 .RE

227 .sp
228 .ne 2
229 .na
230 \fB\&.SH DESCRIPTION\fR
231 .ad
232 .RS 23n
167 A narrative overview of the command or function's external behavior. This
168 includes how it interacts with files or data, and how it handles the standard
169 input, standard output and standard error. Internals and implementation details
170 are normally omitted. This section attempts to provide a succinct overview in
171 answer to the question, "what does it do?"
172 .Lp
238 .sp
173 Literal text from the synopsis appears in constant width, as do literal
174 filenames and references to items that appear elsewhere in the reference
175 manuals. Arguments are italicized.
176 .Lp
242 .sp
177 If a command interprets either subcommands or an input grammar, its command
178 interface or input grammar is normally described in a
179 .Nm USAGE
180 section, which follows the
181 .Nm OPTIONS
182 section. The
183 .Nm DESCRIPTION
184 section only
244 interface or input grammar is normally described in a \fBUSAGE\fR section,

```

```

245 which follows the \fBOPTIONS\fr section. The \fBDESCRIPTION\fr section only
185 describes the behavior of the command itself, not that of subcommands.
186 .
187 .It Nm \&.SH OPTIONS
188 .
247 .RE

249 .sp
250 .ne 2
251 .na
252 \fB\&.SH OPTIONS\fr
253 .ad
254 .RS 23n
189 The list of options along with a description of how each affects the command's
190 operation.
191 .
192 .It Nm \&.SH RETURN VALUES
193 .
257 .RE

259 .sp
260 .ne 2
261 .na
262 \fB\&.SH RETURN VALUES\fr
263 .ad
264 .RS 23n
194 A list of the values the library routine will return to the calling program
195 and the conditions that cause these values to be returned.
196 .
197 .It Nm \&.SH EXIT STATUS
198 .
267 .RE

269 .sp
270 .ne 2
271 .na
272 \fB\&.SH EXIT STATUS\fr
273 .ad
274 .RS 23n
199 A list of the values the utility will return to the calling program or shell,
200 and the conditions that cause these values to be returned.
201 .
202 .It Nm \&.SH FILES
203 .
277 .RE

279 .sp
280 .ne 2
281 .na
282 \fB\&.SH FILES\fr
283 .ad
284 .RS 23n
204 A list of files associated with the command or function.
205 .
206 .It Nm \&.SH SEE ALSO
207 .
286 .RE

288 .sp
289 .ne 2
290 .na
291 \fB\&.SH SEE ALSO\fr
292 .ad
293 .RS 23n
208 A comma-separated list of related manual pages, followed by references to other
209 published materials.

```

```

210 .
211 .It Nm \&.SH DIAGNOSTICS
212 .
296 .RE

298 .sp
299 .ne 2
300 .na
301 \fB\&.SH DIAGNOSTICS\fr
302 .ad
303 .RS 23n
213 A list of diagnostic messages and an explanation of each.
214 .
215 .It Nm \&.SH BUGS
216 .
305 .RE

307 .sp
308 .ne 2
309 .na
310 \fB\&.SH BUGS\fr
311 .ad
312 .RS 23n
217 A description of limitations, known defects, and possible problems associated
218 with the command or function.
219 .El
220 .Sh FILES
221 .Pa /usr/share/man/whatis
222 .Sh NOTES
223 The
224 .Nm
225 package should not be used for new documentation. The
226 .Xr mdoc 5 ,
227 package is preferred, as it uses semantic markup rather than physical markup.
228 .Sh CODE SET INDEPENDENCE
229 When processed with
230 .Xr mandoc 1 ,
231 this package is Code Set Independent. However, when processed with
232 legacy tools such as
233 .Xr nroff 1
234 and
235 .Xr troff 1 ,
236 the use of multi-byte characters may not be supported.
237 .Sh INTERFACE STABILITY
238 .Nm Obsolete Committed .
239 The
240 .Xr mdoc 5
241 package should be used instead.
242 .Sh SEE ALSO
243 .Xr eqn 1 ,
244 .Xr man 1 ,
245 .Xr mandoc 1 ,
246 .Xr nroff 1 ,
247 .Xr troff 1 ,
248 .Xr tbl 1 ,
249 .Xr whatis 1 ,
250 .Xr mdoc 5 ,
251 .Rs
252 .%A Dale Dougherty and Tim O'Reilly
253 .%B Unix Text Processing
254 .Re
315 .RE

317 .Sh FILES
318 .sp
319 .ne 2

```

```
320 .na
321 \fB\fB/usr/share/lib/tmac/an\fR \fR
322 .ad
323 .RS 27n

325 .RE

327 .sp
328 .ne 2
329 .na
330 \fB\fB/usr/share/man/windex\fR\fR
331 .ad
332 .RS 27n

334 .RE

336 .SH SEE ALSO
337 .sp
338 .LP
339 \fBman\fR(1), \fBnroff\fR(1), \fBtroff\fR(1), \fBwhatismail\fR(1)
340 .sp
341 .LP
342 Dale Dougherty and Tim O'Reilly, \fIUnix\fR \fIText\fR \fIProcessing\fR
```

26074 Sat Jul 19 19:24:38 2014

new/usr/src/man/man5/mandoc_char.5

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 .\"
2 .\" Permission to use, copy, modify, and distribute this software for any
3 .\" purpose with or without fee is hereby granted, provided that the above
4 .\" copyright notice and this permission notice appear in all copies.
5 .\"
6 .\" THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
7 .\" WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
8 .\" MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
9 .\" ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
10 .\" WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
11 .\" ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
12 .\" OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
13 .\"
14 .\"
15 .\" Copyright (c) 2003 Jason McIntyre <jmc@openbsd.org>
16 .\" Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
17 .\" Copyright (c) 2011 Ingo Schwarze <schwarze@openbsd.org>
18 .\" Copyright 2012 Nexenta Systems, Inc. All rights reserved.
19 .\"
20 .Dd Nov 23, 2011
21 .Dt MANDOC_CHAR 5
22 .Os
23 .Sh NAME
24 .Nm mandoc_char
25 .Nd mandoc special characters
26 .Sh DESCRIPTION
27 This page documents the
28 .Xr roff 5
29 escape sequences accepted by
30 .Xr mandoc 1
31 to represent special characters in
32 .Xr mdoc 5
33 and
34 .Xr man 5
35 documents.
36 .Pp
37 The rendering depends on the
38 .Xr mandoc 1
39 output mode; in ASCII output, most characters are completely
40 unintelligible.
41 For that reason, using any of the special characters documented here,
42 except those discussed in the
43 .Sx DESCRIPTION ,
44 is strongly discouraged; they are supported merely for backwards
45 compatibility with existing documents.
46 .Pp
47 In particular, in English manual pages, do not use special-character
48 escape sequences to represent national language characters in author
49 names; instead, provide ASCII transcriptions of the names.
50 .Ss Dashes and Hyphens
51 In typography there are different types of dashes of various width:
52 the hyphen (-),
53 the minus sign (\-),
54 the en-dash (\(en),
55 and the em-dash (\(em).
56 .Pp

```

```

57 Hyphens are used for adjectives;
58 to separate the two parts of a compound word;
59 or to separate a word across two successive lines of text.
60 The hyphen does not need to be escaped:
61 .Bd -unfilled -offset indent
62 blue-eyed
63 lorry-driver
64 .Ed
65 .Pp
66 The mathematical minus sign is used for negative numbers or subtraction.
67 It should be written as
68 .Sq \(e- :
69 .Bd -unfilled -offset indent
70 a = 3 \(e- 1;
71 b = \(e-2;
72 .Ed
73 .Pp
74 The en-dash is used to separate the two elements of a range,
75 or can be used the same way as an em-dash.
76 It should be written as
77 .Sq \(e(en :
78 .Bd -unfilled -offset indent
79 pp. 95\(e(en97.
80 Go away \(e(en or else!
81 .Ed
82 .Pp
83 The em-dash can be used to show an interruption
84 or can be used the same way as colons, semi-colons, or parentheses.
85 It should be written as
86 .Sq \(e(em :
87 .Bd -unfilled -offset indent
88 Three things \(e(em apples, oranges, and bananas.
89 This is not that \(e(em rather, this is that.
90 .Ed
91 .Pp
92 Note:
93 hyphens, minus signs, and en-dashes look identical under normal ASCII output.
94 Other formats, such as PostScript, render them correctly,
95 with differing widths.
96 .Ss Spaces
97 To separate words in normal text, for indenting and alignment
98 in literal context, and when none of the following special cases apply,
99 just use the normal space character
100 .Pq Sq \ .
101 .Pp
102 When filling text, lines may be broken between words, i.e. at space
103 characters.
104 To prevent a line break between two particular words,
105 use the non-breaking space escape sequence
106 .Pq Sq \(e~
107 instead of the normal space character.
108 For example, the input string
109 .Dq number\(e-1
110 will be kept together as
111 .Dq number\~1
112 on the same output line.
113 .Pp
114 On request and macro lines, the normal space character serves as an
115 argument delimiter.
116 To include whitespace into arguments, quoting is usually the best choice.
117 In some cases, using either the non-breaking
118 .Pq Sq \(e~
119 or the breaking
120 .Pq Sq \(e\ \&
121 space escape sequence may be preferable.
122 To escape macro names and to protect whitespace at the end

```



```

123 of input lines, the zero-width space
124 .Pq Sq \e&
125 is often useful.
126 For example, in
127 .Xr mdoc 5 ,
128 a normal space character can be displayed in single quotes in either
129 of the following ways:
130 .Pp
131 .Dl .Sq \(\dq \(\dq
132 .Dl .Sq \e \e&
133 .Ss Quotes
134 On request and macro lines, the double-quote character
135 .Pq Sq \(\dq
136 is handled specially to allow quoting.
137 One way to prevent this special handling is by using the
138 .Sq \e(dq
139 escape sequence.
140 .Pp
141 Note that on text lines, literal double-quote characters can be used
142 verbatim.
143 All other quote-like characters can be used verbatim as well,
144 even on request and macro lines.
145 .Ss Periods
146 The period
147 .Pq Sq \&.
148 is handled specially at the beginning of an input line,
149 where it introduces a
150 .Xr roff 5
151 request or a macro, and when appearing alone as a macro argument in
152 .Xr mdoc 5 .
153 In such situations, prepend a zero-width space
154 .Pq Sq \e&.
155 to make it behave like normal text.
156 .Pp
157 Do not use the
158 .Sq \e.
159 escape sequence.
160 It does not prevent special handling of the period.
161 .Ss Backslashes
162 To include a literal backslash
163 .Pq Sq \e
164 into the output, use the
165 .Pq Sq \ee
166 escape sequence.
167 .Pp
168 Note that doubling it
169 .Pq Sq \e\e
170 is not the right way to output a backslash.
171 Because
172 .Xr mandoc 1
173 does not implement full
174 .Xr roff 5
175 functionality, it may work with
176 .Xr mandoc 1 ,
177 but it may have weird effects on complete
178 .Xr roff 5
179 implementations.
180 .Sh SPECIAL CHARACTERS
181 Special characters are encoded as
182 .Sq \eX
183 .Pq for a one-character escape ,
184 .Sq \e(XX
185 .Pq two-character ,
186 and
187 .Sq \e[N]
188 .Pq N-character .

```

```

189 For details, see the
190 .Em Special Characters
191 subsection of the
192 .Xr roff 5
193 manual.
194 .Pp
195 Spacing:
196 .Bl -column "Input" "Description" -offset indent -compact
197 .It Em Input Ta Em Description
198 .It \e~ Ta non-breaking, non-collapsing space
199 .It \e Ta breaking, non-collapsing n-width space
200 .It \e^ Ta zero-width space
201 .It \e% Ta zero-width space
202 .It \e& Ta zero-width space
203 .It \e| Ta zero-width space
204 .It \e0 Ta breaking, non-collapsing digit-width space
205 .It \ec Ta removes any trailing space (if applicable)
206 .El
207 .Pp
208 Lines:
209 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
210 .It Em Input Ta Em Rendered Ta Em Description
211 .It \e(ba Ta \(\ba Ta bar
212 .It \e(br Ta \(\br Ta box rule
213 .It \e(ul Ta \(\ul Ta underscore
214 .It \e(rl Ta \(\rl Ta overline
215 .It \e(bb Ta \(\bb Ta broken bar
216 .It \e(sl Ta \(\sl Ta forward slash
217 .It \e(rs Ta \(\rs Ta backward slash
218 .El
219 .Pp
220 Text markers:
221 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
222 .It Em Input Ta Em Rendered Ta Em Description
223 .It \e(ci Ta \(\ci Ta circle
224 .It \e(bu Ta \(\bu Ta bullet
225 .It \e(dd Ta \(\dd Ta double dagger
226 .It \e(dg Ta \(\dg Ta dagger
227 .It \e(lz Ta \(\lz Ta lozenge
228 .It \e(sq Ta \(\sq Ta white square
229 .It \e(ps Ta \(\ps Ta paragraph
230 .It \e(sc Ta \(\sc Ta section
231 .It \e(lh Ta \(\lh Ta left hand
232 .It \e(rh Ta \(\rh Ta right hand
233 .It \e(at Ta \(\at Ta at
234 .It \e(sh Ta \(\sh Ta hash (pound)
235 .It \e(CR Ta \(\CR Ta carriage return
236 .It \e(OK Ta \(\OK Ta check mark
237 .El
238 .Pp
239 Legal symbols:
240 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
241 .It Em Input Ta Em Rendered Ta Em Description
242 .It \e(co Ta \(\co Ta copyright
243 .It \e(rg Ta \(\rg Ta registered
244 .It \e(tm Ta \(\tm Ta trademarked
245 .El
246 .Pp
247 Punctuation:
248 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
249 .It Em Input Ta Em Rendered Ta Em Description
250 .It \e(em Ta \(\em Ta em-dash
251 .It \e(en Ta \(\en Ta en-dash
252 .It \e(hy Ta \(\hy Ta hyphen
253 .It \ee Ta \e Ta back-slash
254 .It \e. Ta \. Ta period

```

```

255 .It \e(r! Ta \r! Ta upside-down exclamation
256 .It \e(r? Ta \r? Ta upside-down question
257 .El
258 .Pp
259 Quotes:
260 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
261 .It Em Input Ta Em Rendered Ta Em Description
262 .It \e(Bq Ta \Bq Ta right low double-quote
263 .It \e(bq Ta \bq Ta right low single-quote
264 .It \e(lq Ta \lq Ta left double-quote
265 .It \e(rq Ta \rq Ta right double-quote
266 .It \e(oq Ta \oq Ta left single-quote
267 .It \e(cq Ta \cq Ta right single-quote
268 .It \e(aq Ta \aq Ta apostrophe quote (text)
269 .It \e(dq Ta \dq Ta double quote (text)
270 .It \e(Fo Ta \Fo Ta left guillemet
271 .It \e(Fc Ta \Fc Ta right guillemet
272 .It \e(fo Ta \fo Ta left single guillemet
273 .It \e(fc Ta \fc Ta right single guillemet
274 .El
275 .Pp
276 Brackets:
277 .Bl -column "xxbracketrightbpx" Rendered Description -offset indent -compact
278 .It Em Input Ta Em Rendered Ta Em Description
279 .It \e(lB Ta \lB Ta left bracket
280 .It \e(rB Ta \rB Ta right bracket
281 .It \e(lC Ta \lC Ta left brace
282 .It \e(rC Ta \rC Ta right brace
283 .It \e(la Ta \la Ta left angle
284 .It \e(ra Ta \ra Ta right angle
285 .It \e(bv Ta \bv Ta brace extension
286 .It \e[braceex] Ta \[braceex] Ta brace extension
287 .It \e[bracketlefttp] Ta \[bracketlefttp] Ta top-left hooked bracket
288 .It \e[bracketleftbp] Ta \[bracketleftbp] Ta bottom-left hooked bracket
289 .It \e[bracketleftex] Ta \[bracketleftex] Ta left hooked bracket extension
290 .It \e[bracketrighttp] Ta \[bracketrighttp] Ta top-right hooked bracket
291 .It \e[bracketrightbp] Ta \[bracketrightbp] Ta bottom-right hooked bracket
292 .It \e[bracketrightex] Ta \[bracketrightex] Ta right hooked bracket extension
293 .It \e(lt Ta \lt Ta top-left hooked brace
294 .It \e[bracelefttp] Ta \[bracelefttp] Ta top-left hooked brace
295 .It \e(lk Ta \lk Ta mid-left hooked brace
296 .It \e[braceleftmid] Ta \[braceleftmid] Ta mid-left hooked brace
297 .It \e(lb Ta \lb Ta bottom-left hooked brace
298 .It \e[braceleftbp] Ta \[braceleftbp] Ta bottom-left hooked brace
299 .It \e[braceleftex] Ta \[braceleftex] Ta left hooked brace extension
300 .It \e(rt Ta \rt Ta top-left hooked brace
301 .It \e[bracerighttp] Ta \[bracerighttp] Ta top-right hooked brace
302 .It \e(rk Ta \rk Ta mid-right hooked brace
303 .It \e[bracerightmid] Ta \[bracerightmid] Ta mid-right hooked brace
304 .It \e(rb Ta \rb Ta bottom-right hooked brace
305 .It \e[bracerightbp] Ta \[bracerightbp] Ta bottom-right hooked brace
306 .It \e[bracerightex] Ta \[bracerightex] Ta right hooked brace extension
307 .It \e[parenlefttp] Ta \[parenlefttp] Ta top-left hooked parenthesis
308 .It \e[parenleftbp] Ta \[parenleftbp] Ta bottom-left hooked parenthesis
309 .It \e[parenleftex] Ta \[parenleftex] Ta left hooked parenthesis extension
310 .It \e[parenrighttp] Ta \[parenrighttp] Ta top-right hooked parenthesis
311 .It \e[parenrightbp] Ta \[parenrightbp] Ta bottom-right hooked parenthesis
312 .It \e[parenrightex] Ta \[parenrightex] Ta right hooked parenthesis extension
313 .El
314 .Pp
315 Arrows:
316 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
317 .It Em Input Ta Em Rendered Ta Em Description
318 .It \e(<- Ta \<- Ta left arrow
319 .It \e(-> Ta \-> Ta right arrow
320 .It \e(<> Ta \<> Ta left-right arrow

```

```

321 .It \e(da Ta \da Ta down arrow
322 .It \e(ua Ta \ua Ta up arrow
323 .It \e(va Ta \va Ta up-down arrow
324 .It \e(lA Ta \lA Ta left double-arrow
325 .It \e(rA Ta \rA Ta right double-arrow
326 .It \e(hA Ta \hA Ta left-right double-arrow
327 .It \e(uA Ta \uA Ta up double-arrow
328 .It \e(dA Ta \dA Ta down double-arrow
329 .It \e(vA Ta \vA Ta up-down double-arrow
330 .El
331 .Pp
332 Logical:
333 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
334 .It Em Input Ta Em Rendered Ta Em Description
335 .It \e(AN Ta \AN Ta logical and
336 .It \e(OR Ta \OR Ta logical or
337 .It \e(no Ta \no Ta logical not
338 .It \e[tno] Ta \[tno] Ta logical not (text)
339 .It \e(te Ta \te Ta existential quantifier
340 .It \e(fa Ta \fa Ta universal quantifier
341 .It \e(st Ta \st Ta such that
342 .It \e(tf Ta \tf Ta therefore
343 .It \e(3d Ta \3d Ta therefore
344 .It \e(or Ta \or Ta bitwise or
345 .El
346 .Pp
347 Mathematical:
348 .Bl -column "xxcoproductxxx" "Rendered" "Description" -offset indent -compact
349 .It Em Input Ta Em Rendered Ta Em Description
350 .It \e(pl Ta \pl Ta plus
351 .It \e(mi Ta \mi Ta minus
352 .It \e- Ta \- Ta minus (text)
353 .It \e(-+ Ta \(-+ Ta minus-plus
354 .It \e(+-) Ta \+(-) Ta plus-minus
355 .It \e[+-] Ta \[+-] Ta plus-minus (text)
356 .It \e(pc Ta \pc Ta centre-dot
357 .It \e(mu Ta \mu Ta multiply
358 .It \e[tmu] Ta \[tmu] Ta multiply (text)
359 .It \e(c* Ta \c* Ta circle-multiply
360 .It \e(c+ Ta \c+ Ta circle-plus
361 .It \e(di Ta \di Ta divide
362 .It \e[tdi] Ta \[tdi] Ta divide (text)
363 .It \e(f/ Ta \f/ Ta fraction
364 .It \e(** Ta \(** Ta asterisk
365 .It \e(<= Ta \<= Ta less-than-equal
366 .It \e(>= Ta \>= Ta greater-than-equal
367 .It \e(<< Ta \<< Ta much less
368 .It \e(>> Ta \>> Ta much greater
369 .It \e(= Ta \= Ta equal
370 .It \e(!= Ta \!= Ta not equal
371 .It \e(= Ta \= Ta equivalent
372 .It \e(≠ Ta \≠ Ta not equivalent
373 .It \e(≈ Ta \≈ Ta congruent
374 .It \e(∞ Ta \∞ Ta asymptotically congruent
375 .It \e(≈ Ta \≈ Ta asymptotically similar
376 .It \e(∼ Ta \∼ Ta approximately similar
377 .It \e(≈ Ta \≈ Ta approximately equal
378 .It \e(pt Ta \pt Ta proportionate
379 .It \e(es Ta \es Ta empty set
380 .It \e(mo Ta \mo Ta element
381 .It \e(nm Ta \nm Ta not element
382 .It \e(sb Ta \sb Ta proper subset
383 .It \e(nb Ta \nb Ta not subset
384 .It \e(sp Ta \sp Ta proper superset
385 .It \e(nc Ta \nc Ta not superset
386 .It \e(ib Ta \ib Ta reflexive subset

```

```

387 .It \e(ip Ta \{ip Ta reflexive superset
388 .It \e(ca Ta \{ca Ta intersection
389 .It \e(cu Ta \{cu Ta union
390 .It \e(/_ Ta \{/_ Ta angle
391 .It \e(pp Ta \{pp Ta perpendicular
392 .It \e(is Ta \{is Ta integral
393 .It \e[integral] Ta \[[integral] Ta integral
394 .It \e[sum] Ta \[sum] Ta summation
395 .It \e[product] Ta \[product] Ta product
396 .It \e[coproduct] Ta \[coproduct] Ta coproduct
397 .It \e(gr Ta \{gr Ta gradient
398 .It \e(sr Ta \{sr Ta square root
399 .It \e[sqrt] Ta \[sqrt] Ta square root
400 .It \e(lc Ta \{lc Ta left-ceiling
401 .It \e(rc Ta \{rc Ta right-ceiling
402 .It \e(lf Ta \{lf Ta left-floor
403 .It \e(rf Ta \{rf Ta right-floor
404 .It \e(if Ta \{if Ta infinity
405 .It \e(Ah Ta \{Ah Ta aleph
406 .It \e(Im Ta \{Im Ta imaginary
407 .It \e(Re Ta \{Re Ta real
408 .It \e(pd Ta \{pd Ta partial differential
409 .It \e(-h Ta \{-h Ta Planck constant over 2\(*p
410 .It \e[12] Ta \[[12] Ta one-half
411 .It \e[14] Ta \[[14] Ta one-fourth
412 .It \e[34] Ta \[[34] Ta three-fourths
413 .El
414 .Pp
415 Ligatures:
416 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
417 .It Em Input Ta Em Rendered Ta Em Description
418 .It \e(ff Ta \{ff Ta ff ligature
419 .It \e(fi Ta \{fi Ta fi ligature
420 .It \e(fl Ta \{fl Ta fl ligature
421 .It \e(Fi Ta \{Fi Ta ffi ligature
422 .It \e(Fl Ta \{Fl Ta ffl ligature
423 .It \e(AE Ta \{AE Ta AE
424 .It \e(ae Ta \{ae Ta ae
425 .It \e(OE Ta \{OE Ta OE
426 .It \e(oe Ta \{oe Ta oe
427 .It \e(ss Ta \{ss Ta German eszett
428 .It \e(IJ Ta \{IJ Ta IJ ligature
429 .It \e(ij Ta \{ij Ta ij ligature
430 .El
431 .Pp
432 Accents:
433 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
434 .It Em Input Ta Em Rendered Ta Em Description
435 .It \e(a" Ta \{a" Ta Hungarian umlaut
436 .It \e(a- Ta \{a- Ta macron
437 .It \e(a. Ta \{a. Ta dotted
438 .It \e(a^ Ta \{a^ Ta circumflex
439 .It \e(aa Ta \{aa Ta acute
440 .It \e' Ta \{' Ta acute
441 .It \e(ga Ta \{ga Ta grave
442 .It \e` Ta \{` Ta grave
443 .It \e(ab Ta \{ab Ta breve
444 .It \e(ac Ta \{ac Ta cedilla
445 .It \e(ad Ta \{ad Ta dieresis
446 .It \e(ah Ta \{ah Ta caron
447 .It \e(ao Ta \{ao Ta ring
448 .It \e(a~ Ta \{a~ Ta tilde
449 .It \e(ho Ta \{ho Ta ogonek
450 .It \e(ha Ta \{ha Ta hat (text)
451 .It \e(ti Ta \{ti Ta tilde (text)
452 .El

```

```

453 .Pp
454 Accented letters:
455 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
456 .It Em Input Ta Em Rendered Ta Em Description
457 .It \e('A Ta \{'A Ta acute A
458 .It \e('E Ta \{'E Ta acute E
459 .It \e('I Ta \{'I Ta acute I
460 .It \e('O Ta \{'O Ta acute O
461 .It \e('U Ta \{'U Ta acute U
462 .It \e('a Ta \{'a Ta acute a
463 .It \e('e Ta \{'e Ta acute e
464 .It \e('i Ta \{'i Ta acute i
465 .It \e('o Ta \{'o Ta acute o
466 .It \e('u Ta \{'u Ta acute u
467 .It \e('A Ta \{'A Ta grave A
468 .It \e('E Ta \{'E Ta grave E
469 .It \e('I Ta \{'I Ta grave I
470 .It \e('O Ta \{'O Ta grave O
471 .It \e('U Ta \{'U Ta grave U
472 .It \e('a Ta \{'a Ta grave a
473 .It \e('e Ta \{'e Ta grave e
474 .It \e('i Ta \{'i Ta grave i
475 .It \e('o Ta \{'o Ta grave o
476 .It \e('u Ta \{'u Ta grave u
477 .It \e(~A Ta \{~A Ta tilde A
478 .It \e(~N Ta \{~N Ta tilde N
479 .It \e(~O Ta \{~O Ta tilde O
480 .It \e(~a Ta \{~a Ta tilde a
481 .It \e(~n Ta \{~n Ta tilde n
482 .It \e(~o Ta \{~o Ta tilde o
483 .It \e(:A Ta \{:A Ta dieresis A
484 .It \e(:E Ta \{:E Ta dieresis E
485 .It \e(:I Ta \{:I Ta dieresis I
486 .It \e(:O Ta \{:O Ta dieresis O
487 .It \e(:U Ta \{:U Ta dieresis U
488 .It \e(:a Ta \{:a Ta dieresis a
489 .It \e(:e Ta \{:e Ta dieresis e
490 .It \e(:i Ta \{:i Ta dieresis i
491 .It \e(:o Ta \{:o Ta dieresis o
492 .It \e(:u Ta \{:u Ta dieresis u
493 .It \e(:y Ta \{:y Ta dieresis y
494 .It \e(^A Ta \{^A Ta circumflex A
495 .It \e(^E Ta \{^E Ta circumflex E
496 .It \e(^I Ta \{^I Ta circumflex I
497 .It \e(^O Ta \{^O Ta circumflex O
498 .It \e(^U Ta \{^U Ta circumflex U
499 .It \e(^a Ta \{^a Ta circumflex a
500 .It \e(^e Ta \{^e Ta circumflex e
501 .It \e(^i Ta \{^i Ta circumflex i
502 .It \e(^o Ta \{^o Ta circumflex o
503 .It \e(^u Ta \{^u Ta circumflex u
504 .It \e(/,C Ta \{/ ,C Ta cedilla C
505 .It \e(/,c Ta \{/ ,c Ta cedilla c
506 .It \e(/L Ta \{/L Ta stroke L
507 .It \e(/l Ta \{/l Ta stroke l
508 .It \e(/O Ta \{/O Ta stroke O
509 .It \e(/o Ta \{/o Ta stroke o
510 .It \e(oA Ta \{oA Ta ring A
511 .It \e(oa Ta \{oa Ta ring a
512 .El
513 .Pp
514 Special letters:
515 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
516 .It Em Input Ta Em Rendered Ta Em Description
517 .It \e(-D Ta \{-D Ta Eth
518 .It \e(Sd Ta \{Sd Ta eth

```

```

519 .It \e(TP Ta \TP Ta Thorn
520 .It \e(Tp Ta \Tp Ta thorn
521 .It \e(.i Ta \(.i Ta dotless i
522 .It \e(.j Ta \(.j Ta dotless j
523 .El
524 .Pp
525 Currency:
526 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
527 .It Em Input Ta Em Rendered Ta Em Description
528 .It \e(Do Ta \Do Ta dollar
529 .It \e(ct Ta \ct Ta cent
530 .It \e(Eu Ta \Eu Ta Euro symbol
531 .It \e(eu Ta \eu Ta Euro symbol
532 .It \e(Ye Ta \Ye Ta yen
533 .It \e(Po Ta \Po Ta pound
534 .It \e(Cs Ta \Cs Ta Scandinavian
535 .It \e(Fn Ta \Fn Ta florin
536 .El
537 .Pp
538 Units:
539 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
540 .It Em Input Ta Em Rendered Ta Em Description
541 .It \e(de Ta \de Ta degree
542 .It \e(%0 Ta \(%0 Ta per-thousand
543 .It \e(fm Ta \fm Ta minute
544 .It \e(sd Ta \sd Ta second
545 .It \e(mc Ta \mc Ta micro
546 .El
547 .Pp
548 Greek letters:
549 .Bl -column "Input" "Rendered" "Description" -offset indent -compact
550 .It Em Input Ta Em Rendered Ta Em Description
551 .It \e(*A Ta \(*A Ta Alpha
552 .It \e(*B Ta \(*B Ta Beta
553 .It \e(*G Ta \(*G Ta Gamma
554 .It \e(*D Ta \(*D Ta Delta
555 .It \e(*E Ta \(*E Ta Epsilon
556 .It \e(*Z Ta \(*Z Ta Zeta
557 .It \e(*Y Ta \(*Y Ta Eta
558 .It \e(*H Ta \(*H Ta Theta
559 .It \e(*I Ta \(*I Ta Iota
560 .It \e(*K Ta \(*K Ta Kappa
561 .It \e(*L Ta \(*L Ta Lambda
562 .It \e(*M Ta \(*M Ta Mu
563 .It \e(*N Ta \(*N Ta Nu
564 .It \e(*C Ta \(*C Ta Xi
565 .It \e(*O Ta \(*O Ta Omicron
566 .It \e(*P Ta \(*P Ta Pi
567 .It \e(*R Ta \(*R Ta Rho
568 .It \e(*S Ta \(*S Ta Sigma
569 .It \e(*T Ta \(*T Ta Tau
570 .It \e(*U Ta \(*U Ta Upsilon
571 .It \e(*F Ta \(*F Ta Phi
572 .It \e(*X Ta \(*X Ta Chi
573 .It \e(*Q Ta \(*Q Ta Psi
574 .It \e(*W Ta \(*W Ta Omega
575 .It \e(*a Ta \(*a Ta alpha
576 .It \e(*b Ta \(*b Ta beta
577 .It \e(*g Ta \(*g Ta gamma
578 .It \e(*d Ta \(*d Ta delta
579 .It \e(*e Ta \(*e Ta epsilon
580 .It \e(*z Ta \(*z Ta zeta
581 .It \e(*y Ta \(*y Ta eta
582 .It \e(*h Ta \(*h Ta theta
583 .It \e(*i Ta \(*i Ta iota
584 .It \e(*k Ta \(*k Ta kappa

```

```

585 .It \e(*l Ta \(*l Ta lambda
586 .It \e(*m Ta \(*m Ta mu
587 .It \e(*n Ta \(*n Ta nu
588 .It \e(*c Ta \(*c Ta xi
589 .It \e(*o Ta \(*o Ta omicron
590 .It \e(*p Ta \(*p Ta pi
591 .It \e(*r Ta \(*r Ta rho
592 .It \e(*s Ta \(*s Ta sigma
593 .It \e(*t Ta \(*t Ta tau
594 .It \e(*u Ta \(*u Ta upsilon
595 .It \e(*f Ta \(*f Ta phi
596 .It \e(*x Ta \(*x Ta chi
597 .It \e(*q Ta \(*q Ta psi
598 .It \e(*w Ta \(*w Ta omega
599 .It \e(+h Ta \(+h Ta theta variant
600 .It \e(+f Ta \(+f Ta phi variant
601 .It \e(+p Ta \(+p Ta pi variant
602 .It \e(+e Ta \(+e Ta epsilon variant
603 .It \e(ts Ta \ts Ta sigma terminal
604 .El
605 .Sh PREDEFINED STRINGS
606 Predefined strings are inherited from the macro packages of historical
607 troff implementations.
608 They are
609 .Em not recommended
610 for use, as they differ across implementations.
611 Manuals using these predefined strings are almost certainly not
612 portable.
613 .Pp
614 Their syntax is similar to special characters, using
615 .Sq \e*X
616 .Pq for a one-character escape ,
617 .Sq \e*(XX
618 .Pq two-character ,
619 and
620 .Sq \e*[N]
621 .Pq N-character .
622 For details, see the
623 .Em Predefined Strings
624 subsection of the
625 .Xr roff 5
626 manual.
627 .Bl -column "Input" "Rendered" "Description" -offset indent
628 .It Em Input Ta Em Rendered Ta Em Description
629 .It \e*(Ba Ta \*(Ba Ta vertical bar
630 .It \e*(Ne Ta \*(Ne Ta not equal
631 .It \e*(Ge Ta \*(Ge Ta greater-than-equal
632 .It \e*(Le Ta \*(Le Ta less-than-equal
633 .It \e*(Gt Ta \*(Gt Ta greater-than
634 .It \e*(Lt Ta \*(Lt Ta less-than
635 .It \e*(Pm Ta \*(Pm Ta plus-minus
636 .It \e*(If Ta \*(If Ta infinity
637 .It \e*(Pi Ta \*(Pi Ta pi
638 .It \e*(Na Ta \*(Na Ta NaN
639 .It \e*(Am Ta \*(Am Ta ampersand
640 .It \e*(R Ta \*(R Ta restricted mark
641 .It \e*(Tm Ta \*(Tm Ta trade mark
642 .It \e*(q Ta \*(q Ta double-quote
643 .It \e*(Rq Ta \*(Rq Ta right-double-quote
644 .It \e*(Lq Ta \*(Lq Ta left-double-quote
645 .It \e*(lp Ta \*(lp Ta right-parenthesis
646 .It \e*(rp Ta \*(rp Ta left-parenthesis
647 .It \e*(lq Ta \*(lq Ta left double-quote
648 .It \e*(rq Ta \*(rq Ta right double-quote
649 .It \e*(ua Ta \*(ua Ta up arrow
650 .It \e*(va Ta \*(va Ta up-down arrow

```

```

651 .It \e*(<= Ta \*(<= Ta less-than-equal
652 .It \e*(>= Ta \*(>= Ta greater-than-equal
653 .It \e*(aa Ta \*(aa Ta acute
654 .It \e*(ga Ta \*(ga Ta grave
655 .It \e*(Px Ta \*(Px Ta POSIX standard name
656 .It \e*(Ai Ta \*(Ai Ta ANSI standard name
657 .El
658 .Sh UNICODE CHARACTERS
659 The escape sequence
660 .Pp
661 .Dl \e[uXXXX]
662 .Pp
663 is interpreted as a Unicode codepoint.
664 The codepoint must be in the range above U+0080 and less than U+10FFFF.
665 For compatibility, points must be zero-padded to four characters; if
666 greater than four characters, no zero padding is allowed.
667 Unicode surrogates are not allowed.
668 .\" .Pp
669 .\" Unicode glyphs attenuate to the
670 .\" .Sq \&?
671 .\" character if invalid or not rendered by current output media.
672 .Sh NUMBERED CHARACTERS
673 For backward compatibility with existing manuals,
674 .Xr mandoc 1
675 also supports the
676 .Pp
677 .Dl \eN\{aq Ns Ar number Ns \{aq
678 .Pp
679 escape sequence, inserting the character
680 .Ar number
681 from the current character set into the output.
682 Of course, this is inherently non-portable and is already marked
683 as deprecated in the Heirloom roff manual.
684 For example, do not use \eN'34', use \e(dq, or even the plain
685 .Sq \{dq
686 character where possible.
687 .Sh COMPATIBILITY
688 This section documents compatibility between mandoc and other other
689 troff implementations, at this time limited to GNU troff
690 .Pq Qq groff .
691 .Pp
692 .Bl -dash -compact
693 .It
694 The \eN\{aq\{aq escape sequence is limited to printable characters; in
695 groff, it accepts arbitrary character numbers.
696 .It
697 In
698 .Fl T Ns Cm ascii ,
699 the
700 \e(ss, \e(nm, \e(nb, \e(nc, \e(ib, \e(ip, \e(pp, \e[sum], \e[product],
701 \e[coproduct], \e(gr, \e(\-h, and \e(a. special characters render
702 differently between mandoc and groff.
703 .It
704 In
705 .Fl T Ns Cm html
706 and
707 .Fl T Ns Cm xhtml ,
708 the \e(-=, \e(nb, and \e(nc special characters render differently
709 between mandoc and groff.
710 .It
711 The
712 .Fl T Ns Cm ps
713 and
714 .Fl T Ns Cm pdf
715 modes format like
716 .Fl T Ns Cm ascii

```

```

717 instead of rendering glyphs as in groff.
718 .It
719 The \e[radicalex], \e[sqrtex], and \e(ru special characters have been omitted
720 from mandoc either because they are poorly documented or they have no
721 known representation.
722 .El
723 .Sh SEE ALSO
724 .Xr mandoc 1 ,
725 .Xr man 5 ,
726 .Xr mdoc 5 ,
727 .Xr roff 5
728 .Sh AUTHORS
729 The
730 .Nm
731 manual page was written by
732 .An Kristaps Dzonsons ,
733 .Mt kristaps@bsd.lv .
734 .Sh CAVEATS
735 The
736 .Sq \e*(Ba
737 escape mimics the behaviour of the
738 .Sq \&|
739 character in
740 .Xr mdoc 5 ;
741 thus, if you wish to render a vertical bar with no side effects, use
742 the
743 .Sq \e(ba
744 escape.

```

```

*****
23887 Sat Jul 19 19:24:38 2014
new/usr/src/man/man5/mandoc_roff.5
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****

```

```

1 .\"
2 .\" Permission to use, copy, modify, and distribute this software for any
3 .\" purpose with or without fee is hereby granted, provided that the above
4 .\" copyright notice and this permission notice appear in all copies.
5 .\"
6 .\" THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
7 .\" WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
8 .\" MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
9 .\" ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
10 .\" WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
11 .\" ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
12 .\" OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
13 .\"
14 .\"
15 .\" Copyright (c) 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
16 .\" Copyright (c) 2010, 2011 Ingo Schwarze <schwarze@openbsd.org>
17 .\" Copyright 2012 Nexenta Systems, Inc. All rights reserved.
18 .\" Copyright 2014 Garrett D'Amore <garrett@damore.org>
19 .\"
20 .Dd Jul 13, 2014
21 .Dt MANDOC_ROFF 5
22 .Os
23 .Sh NAME
24 .Nm mandoc_roff
25 .Nd roff language reference for mandoc
26 .Sh DESCRIPTION
27 The
28 .Nm roff
29 language is a general purpose text formatting language.
30 Since traditional implementations of the
31 .Xr mdoc 5
32 and
33 .Xr man 5
34 manual formatting languages are based on it,
35 many real-world manuals use small numbers of
36 .Nm
37 requests intermixed with their
38 .Xr mdoc 5
39 or
40 .Xr man 5
41 code.
42 To properly format such manuals, the
43 .Xr mandoc 1
44 utility supports a tiny subset of
45 .Nm
46 requests.
47 Only these requests supported by
48 .Xr mandoc 1
49 are documented in the present manual,
50 together with the basic language syntax shared by
51 .Nm ,
52 .Xr mdoc 5 ,
53 and
54 .Xr man 5 .
55 For complete
56 .Nm

```

```

57 manuals, consult the
58 .Sx SEE ALSO
59 section.
60 .Pp
61 Input lines beginning with the control character
62 .Sq \&.
63 are parsed for requests and macros.
64 Such lines are called
65 .Dq request lines
66 or
67 .Dq macro lines ,
68 respectively.
69 Requests change the processing state and manipulate the formatting;
70 some macros also define the document structure and produce formatted
71 output.
72 The single quote
73 .Pq Qq \{aq
74 is accepted as an alternative control character,
75 treated by
76 .Xr mandoc 1
77 just like
78 .Ql \&.
79 .Pp
80 Lines not beginning with control characters are called
81 .Dq text lines .
82 They provide free-form text to be printed; the formatting of the text
83 depends on the respective processing context.
84 .Sh LANGUAGE SYNTAX
85 .Nm
86 documents may contain only graphable 7-bit ASCII characters, the space
87 character, and, in certain circumstances, the tab character.
88 The back-space character
89 .Sq \e
90 indicates the start of an escape sequence for
91 .Sx Comments ,
92 .Sx Special Characters ,
93 .Sx Predefined Strings ,
94 and
95 user-defined strings defined using the
96 .Sx ds
97 request.
98 .Ss Comments
99 Text following an escaped double-quote
100 .Sq \e\{dq ,
101 whether in a request, macro, or text line, is ignored to the end of the line.
102 A request line beginning with a control character and comment escape
103 .Sq \&.\e\{dq
104 is also ignored.
105 Furthermore, request lines with only a control character and optional
106 trailing whitespace are stripped from input.
107 .Pp
108 Examples:
109 .Bd -literal -offset indent -compact
110 \&.\e\{dq This is a comment line.
111 \&.\e\{dq The next line is ignored:
112 \&.
113 \&.Sh EXAMPLES \e\{dq This is a comment, too.
114 \&example text \e\{dq And so is this.
115 .Ed
116 .Ss Special Characters
117 Special characters are used to encode special glyphs and are rendered
118 differently across output media.
119 They may occur in request, macro, and text lines.
120 Sequences begin with the escape character
121 .Sq \e
122 followed by either an open-parenthesis

```

```

123 .Sq \&(
124 for two-character sequences; an open-bracket
125 .Sq \&[
126 for n-character sequences (terminated at a close-bracket
127 .Sq \&] );
128 or a single one character sequence.
129 .Pp
130 Examples:
131 .Bl -tag -width Ds -offset indent -compact
132 .It Li \e(em
133 Two-letter em dash escape.
134 .It Li \ee
135 One-letter backslash escape.
136 .El
137 .Pp
138 See
139 .Xr mandoc_char 5
140 for a complete list.
141 .Ss Text Decoration
142 Terms may be text-decorated using the
143 .Sq \ef
144 escape followed by an indicator: B (bold), I (italic), R (regular), or P
145 (revert to previous mode).
146 A numerical representation 3, 2, or 1 (bold, italic, and regular,
147 respectively) may be used instead.
148 The indicator or numerical representative may be preceded by C
149 (constant-width), which is ignored.
150 .Pp
151 Examples:
152 .Bl -tag -width Ds -offset indent -compact
153 .It Li \efBbold\efR
154 Write in bold, then switch to regular font mode.
155 .It Li \efIitalic\efP
156 Write in italic, then return to previous font mode.
157 .El
158 .Pp
159 Text decoration is
160 .Em not
161 recommended for
162 .Xr mdoc 5 ,
163 which encourages semantic annotation.
164 .Ss Predefined Strings
165 Predefined strings, like
166 .Sx Special Characters ,
167 mark special output glyphs.
168 Predefined strings are escaped with the slash-asterisk,
169 .Sq \e* :
170 single-character
171 .Sq \e*X ,
172 two-character
173 .Sq \e*(XX ,
174 and N-character
175 .Sq \e*[N] .
176 .Pp
177 Examples:
178 .Bl -tag -width Ds -offset indent -compact
179 .It Li \e*(Am
180 Two-letter ampersand predefined string.
181 .It Li \e*q
182 One-letter double-quote predefined string.
183 .El
184 .Pp
185 Predefined strings are not recommended for use,
186 as they differ across implementations.
187 Those supported by
188 .Xr mandoc 1

```

```

189 are listed in
190 .Xr mandoc_char 5 .
191 Manuals using these predefined strings are almost certainly not portable.
192 .Ss Whitespace
193 Whitespace consists of the space character.
194 In text lines, whitespace is preserved within a line.
195 In request and macro lines, whitespace delimits arguments and is discarded.
196 .Pp
197 Unescaped trailing spaces are stripped from text line input unless in a
198 literal context.
199 In general, trailing whitespace on any input line is discouraged for
200 reasons of portability.
201 In the rare case that a blank character is needed at the end of an
202 input line, it may be forced by
203 .Sq \e\ \& .
204 .Pp
205 Literal space characters can be produced in the output
206 using escape sequences.
207 In macro lines, they can also be included in arguments using quotation; see
208 .Sx MACRO SYNTAX
209 for details.
210 .Pp
211 Blank text lines, which may include whitespace, are only permitted
212 within literal contexts.
213 If the first character of a text line is a space, that line is printed
214 with a leading newline.
215 .Ss Scaling Widths
216 Many requests and macros support scaled widths for their arguments.
217 The syntax for a scaled width is
218 .Sq Li [+]?[0-9]*.[0-9]*[:unit:] ,
219 where a decimal must be preceded or followed by at least one digit.
220 Negative numbers, while accepted, are truncated to zero.
221 .Pp
222 The following scaling units are accepted:
223 .Pp
224 .Bl -tag -width Ds -offset indent -compact
225 .It c
226 centimetre
227 .It i
228 inch
229 .It P
230 pica (~1/6 inch)
231 .It p
232 point (~1/72 inch)
233 .It f
234 synonym for
235 .Sq u
236 .It v
237 default vertical span
238 .It m
239 width of rendered
240 .Sq m
241 .Pq em
242 character
243 .It n
244 width of rendered
245 .Sq n
246 .Pq en
247 character
248 .It u
249 default horizontal span
250 .It M
251 mini-em (~1/100 em)
252 .El
253 .Pp
254 Using anything other than

```

```

255 .Sq m ,
256 .Sq n ,
257 .Sq u ,
258 or
259 .Sq v
260 is necessarily non-portable across output media.
261 See
262 .Sx COMPATIBILITY .
263 .Pp
264 If a scaling unit is not provided, the numerical value is interpreted
265 under the default rules of
266 .Sq v
267 for vertical spaces and
268 .Sq u
269 for horizontal ones.
270 .Pp
271 Examples:
272 .Bl -tag -width ".Bl -tag -width 2i" -offset indent -compact
273 .It Li \&.Bl -tag -width 2i
274 two-inch tagged list indentation in
275 .Xr mdoc 5
276 .It Li \&.HP 2i
277 two-inch tagged list indentation in
278 .Xr man 5
279 .It Li \&.sp 2v
280 two vertical spaces
281 .El
282 .Ss Sentence Spacing
283 Each sentence should terminate at the end of an input line.
284 By doing this, a formatter will be able to apply the proper amount of
285 spacing after the end of sentence (unescaped) period, exclamation mark,
286 or question mark followed by zero or more non-sentence closing
287 delimiters
288 .Po
289 .Sq \& ) ,
290 .Sq \& ] ,
291 .Sq \& ' ,
292 .Sq \& "
293 .Pc .
294 .Pp
295 The proper spacing is also intelligently preserved if a sentence ends at
296 the boundary of a macro line.
297 .Pp
298 Examples:
299 .Bd -literal -offset indent -compact
300 Do not end sentences mid-line like this. Instead,
301 end a sentence like this.
302 A macro would end like this:
303 \&.Xr mandoc 1 \&.
304 .Ed
305 .Sh REQUEST SYNTAX
306 A request or macro line consists of:
307 .Pp
308 .Bl -enum -compact
309 .It
310 the control character
311 .Sq \&.
312 or
313 .Sq \{(aq
314 at the beginning of the line,
315 .It
316 optionally an arbitrary amount of whitespace,
317 .It
318 the name of the request or the macro, which is one word of arbitrary
319 length, terminated by whitespace,
320 .It

```

```

321 and zero or more arguments delimited by whitespace.
322 .El
323 .Pp
324 Thus, the following request lines are all equivalent:
325 .Bd -literal -offset indent
326 \&.ig end
327 \&.ig end
328 \&. ig end
329 .Ed
330 .Sh MACRO SYNTAX
331 Macros are provided by the
332 .Xr mdoc 5
333 and
334 .Xr man 5
335 languages and can be defined by the
336 .Sx \&de
337 request.
338 When called, they follow the same syntax as requests, except that
339 macro arguments may optionally be quoted by enclosing them
340 in double quote characters
341 .Pq Sq \{(dq .
342 Quoted text, even if it contains whitespace or would cause
343 a macro invocation when unquoted, is always considered literal text.
344 Inside quoted text, pairs of double quote characters
345 .Pq Sq Qq
346 resolve to single double quote characters.
347 .Pp
348 To be recognised as the beginning of a quoted argument, the opening
349 quote character must be preceded by a space character.
350 A quoted argument extends to the next double quote character that is not
351 part of a pair, or to the end of the input line, whichever comes earlier.
352 Leaving out the terminating double quote character at the end of the line
353 is discouraged.
354 For clarity, if more arguments follow on the same input line,
355 it is recommended to follow the terminating double quote character
356 by a space character; in case the next character after the terminating
357 double quote character is anything else, it is regarded as the beginning
358 of the next, unquoted argument.
359 .Pp
360 Both in quoted and unquoted arguments, pairs of backslashes
361 .Pq Sq \e\e
362 resolve to single backslashes.
363 In unquoted arguments, space characters can alternatively be included
364 by preceding them with a backslash
365 .Pq Sq \e\~ ,
366 but quoting is usually better for clarity.
367 .Pp
368 Examples:
369 .Bl -tag -width Ds -offset indent -compact
370 .It Li .Fn strlen \{(dqconst char *s\{(dq
371 Group arguments
372 .Qq const char *s
373 into one function argument.
374 If unspecified,
375 .Qq const ,
376 .Qq char ,
377 and
378 .Qq *s
379 would be considered separate arguments.
380 .It Li .Op \{(dqfl a\{(dq
381 Consider
382 .Qq \&Fl a
383 as literal text instead of a flag macro.
384 .El
385 .Sh REQUEST REFERENCE
386 The

```



```

387 .Xr mandoc 1
388 .Nm
389 parser recognises the following requests.
390 Note that the
391 .Nm
392 language defines many more requests not implemented in
393 .Xr mandoc 1 .
394 .Ss \&ad
395 Set line adjustment mode.
396 This line-scoped request is intended to have one argument to select
397 normal, left, right, or centre adjustment for subsequent text.
398 Currently, it is ignored including its arguments,
399 and the number of arguments is not checked.
400 .Ss \&am
401 Append to a macro definition.
402 The syntax of this request is the same as that of
403 .Sx \&de .
404 It is currently ignored by
405 .Xr mandoc 1 ,
406 as are its children.
407 .Ss \&ami
408 Append to a macro definition, specifying the macro name indirectly.
409 The syntax of this request is the same as that of
410 .Sx \&dei .
411 It is currently ignored by
412 .Xr mandoc 1 ,
413 as are its children.
414 .Ss \&aml
415 Append to a macro definition, switching roff compatibility mode off
416 during macro execution.
417 The syntax of this request is the same as that of
418 .Sx \&del .
419 It is currently ignored by
420 .Xr mandoc 1 ,
421 as are its children.
422 .Ss \&de
423 Define a
424 .Nm
425 macro.
426 Its syntax can be either
427 .Bd -literal -offset indent
428 .Pf . Cm \&de Ar name
429 .Ar macro definition
430 \&..
431 .Ed
432 .Pp
433 or
434 .Bd -literal -offset indent
435 .Pf . Cm \&de Ar name Ar end
436 .Ar macro definition
437 .Pf . Ar end
438 .Ed
439 .Pp
440 Both forms define or redefine the macro
441 .Ar name
442 to represent the
443 .Ar macro definition ,
444 which may consist of one or more input lines, including the newline
445 characters terminating each line, optionally containing calls to
446 .Nm
447 requests,
448 .Nm
449 macros or high-level macros like
450 .Xr man 5
451 or
452 .Xr mdoc 5

```

```

453 macros, whichever applies to the document in question.
454 .Pp
455 Specifying a custom
456 .Ar end
457 macro works in the same way as for
458 .Sx \&ig ;
459 namely, the call to
460 .Sq Pf . Ar end
461 first ends the
462 .Ar macro definition ,
463 and after that, it is also evaluated as a
464 .Nm
465 request or
466 .Nm
467 macro, but not as a high-level macro.
468 .Pp
469 The macro can be invoked later using the syntax
470 .Pp
471 .Dl Pf . Ar name Op Ar argument Op Ar argument ...
472 .Pp
473 Regarding argument parsing, see
474 .Sx MACRO SYNTAX
475 above.
476 .Pp
477 The line invoking the macro will be replaced
478 in the input stream by the
479 .Ar macro definition ,
480 replacing all occurrences of
481 .No \e\e$ Ns Ar N ,
482 where
483 .Ar N
484 is a digit, by the
485 .Ar N Ns th Ar argument .
486 For example,
487 .Bd -literal -offset indent
488 \&.de ZN
489 \efI\e^\e\e$1\e^\efP\e\e$2
490 \&..
491 \&.ZN XtFree .
492 .Ed
493 .Pp
494 produces
495 .Pp
496 .Dl \efI\e^XtFree\e^\efP.
497 .Pp
498 in the input stream, and thus in the output: \fI^XtFree^\fP.
499 .Pp
500 Since macros and user-defined strings share a common string table,
501 defining a macro
502 .Ar name
503 clobbers the user-defined string
504 .Ar name ,
505 and the
506 .Ar macro definition
507 can also be printed using the
508 .Sq \e*
509 string interpolation syntax described below
510 .Sx ds ,
511 but this is rarely useful because every macro definition contains at least
512 one explicit newline character.
513 .Pp
514 In order to prevent endless recursion, both groff and
515 .Xr mandoc 1
516 limit the stack depth for expanding macros and strings
517 to a large, but finite number.
518 Do not rely on the exact value of this limit.

```

```

519 .Ss \&dei
520 Define a
521 .Nm
522 macro, specifying the macro name indirectly.
523 The syntax of this request is the same as that of
524 .Sx \&de .
525 It is currently ignored by
526 .Xr mandoc 1 ,
527 as are its children.
528 .Ss \&del
529 Define a
530 .Nm
531 macro that will be executed with
532 .Nm
533 compatibility mode switched off during macro execution.
534 This is a GNU extension not available in traditional
535 .Nm
536 implementations and not even in older versions of groff.
537 Since
538 .Xr mandoc 1
539 does not implement
540 .Nm
541 compatibility mode at all, it handles this request as an alias for
542 .Sx \&de .
543 .Ss \&ds
544 Define a user-defined string.
545 Its syntax is as follows:
546 .Pp
547 .Dl Pf . Cm \&ds Ar name Oo \{(dq Oc Ns Ar string
548 .Pp
549 The
550 .Ar name
551 and
552 .Ar string
553 arguments are space-separated.
554 If the
555 .Ar string
556 begins with a double-quote character, that character will not be part
557 of the string.
558 All remaining characters on the input line form the
559 .Ar string ,
560 including whitespace and double-quote characters, even trailing ones.
561 .Pp
562 The
563 .Ar string
564 can be interpolated into subsequent text by using
565 .No \e* Ns Bq Ar name
566 for a
567 .Ar name
568 of arbitrary length, or \e*(NN or \e*N if the length of
569 .Ar name
570 is two or one characters, respectively.
571 Interpolation can be prevented by escaping the leading backslash;
572 that is, an asterisk preceded by an even number of backslashes
573 does not trigger string interpolation.
574 .Pp
575 Since user-defined strings and macros share a common string table,
576 defining a string
577 .Ar name
578 clobbers the macro
579 .Ar name ,
580 and the
581 .Ar name
582 used for defining a string can also be invoked as a macro,
583 in which case the following input line will be appended to the
584 .Ar string ,

```

```

585 forming a new input line passed to the
586 .Nm
587 parser.
588 For example,
589 .Bd -literal -offset indent
590 \&.ds badidea .S
591 \&.badidea
592 H SYNOPSIS
593 .Ed
594 .Pp
595 invokes the
596 .Cm SH
597 macro when used in a
598 .Xr man 5
599 document.
600 Such abuse is of course strongly discouraged.
601 .Ss \&el
602 The
603 .Qq else
604 half of an if/else conditional.
605 Pops a result off the stack of conditional evaluations pushed by
606 .Sx \&ie
607 and uses it as its conditional.
608 If no stack entries are present (e.g., due to no prior
609 .Sx \&ie
610 calls)
611 then false is assumed.
612 The syntax of this request is similar to
613 .Sx \&if
614 except that the conditional is missing.
615 .Ss \&EN
616 End an equation block.
617 See
618 .Sx \&EQ .
619 .Ss \&EQ
620 Begin an equation block.
621 See
622 .Xr eqn 5
623 for a description of the equation language.
624 .Ss \&hy
625 Set automatic hyphenation mode.
626 This line-scoped request is currently ignored.
627 .Ss \&ie
628 The
629 .Qq if
630 half of an if/else conditional.
631 The result of the conditional is pushed into a stack used by subsequent
632 invocations of
633 .Sx \&el ,
634 which may be separated by any intervening input (or not exist at all).
635 Its syntax is equivalent to
636 .Sx \&if .
637 .Ss \&if
638 Begins a conditional.
639 Right now, the conditional evaluates to true
640 if and only if it starts with the letter
641 .Sy n ,
642 indicating processing in nroff style as opposed to troff style.
643 If a conditional is false, its children are not processed, but are
644 syntactically interpreted to preserve the integrity of the input
645 document.
646 Thus,
647 .Pp
648 .Dl \&.if t .ig
649 .Pp
650 will discard the

```

```

651 .Sq \&.ig ,
652 which may lead to interesting results, but
653 .Pp
654 .Dl \&.if t .if t \e{\e
655 .Pp
656 will continue to syntactically interpret to the block close of the final
657 conditional.
658 Sub-conditionals, in this case, obviously inherit the truth value of
659 the parent.
660 This request has the following syntax:
661 .Bd -literal -offset indent
662 \&.if COND \e{\e
663 BODY...
664 \&.\e}
665 .Ed
666 .Bd -literal -offset indent
667 \&.if COND \e{ BODY
668 BODY... \e}
669 .Ed
670 .Bd -literal -offset indent
671 \&.if COND \e{ BODY
672 BODY...
673 \&.\e}
674 .Ed
675 .Bd -literal -offset indent
676 \&.if COND \e
677 BODY
678 .Ed
679 .Pp
680 COND is a conditional statement.
681 roff allows for complicated conditionals; mandoc is much simpler.
682 At this time, mandoc supports only
683 .Sq n ,
684 evaluating to true;
685 and
686 .Sq t ,
687 .Sq e ,
688 and
689 .Sq o ,
690 evaluating to false.
691 All other invocations are read up to the next end of line or space and
692 evaluate as false.
693 .Pp
694 If the BODY section is begun by an escaped brace
695 .Sq \e{ ,
696 scope continues until a closing-brace escape sequence
697 .Sq \. \e} .
698 If the BODY is not enclosed in braces, scope continues until
699 the end of the line.
700 If the COND is followed by a BODY on the same line, whether after a
701 brace or not, then requests and macros
702 .Em must
703 begin with a control character.
704 It is generally more intuitive, in this case, to write
705 .Bd -literal -offset indent
706 \&.if COND \e{\e
707 \&.foo
708 bar
709 \&.\e}
710 .Ed
711 .Pp
712 than having the request or macro follow as
713 .Pp
714 .Dl \&.if COND \e{ .foo
715 .Pp
716 The scope of a conditional is always parsed, but only executed if the

```

```

717 conditional evaluates to true.
718 .Pp
719 Note that the
720 .Sq \e}
721 is converted into a zero-width escape sequence if not passed as a
722 standalone macro
723 .Sq \&.\e} .
724 For example,
725 .Pp
726 .Dl \&.Fl a \e} b
727 .Pp
728 will result in
729 .Sq \e}
730 being considered an argument of the
731 .Sq \&Fl
732 macro.
733 .Ss \&ig
734 Ignore input.
735 Its syntax can be either
736 .Bd -literal -offset indent
737 .Pf . Cm \&ig
738 .Ar ignored text
739 \&..
740 .Ed
741 .Pp
742 or
743 .Bd -literal -offset indent
744 .Pf . Cm \&ig Ar end
745 .Ar ignored text
746 .Pf . Ar end
747 .Ed
748 .Pp
749 In the first case, input is ignored until a
750 .Sq \&..
751 request is encountered on its own line.
752 In the second case, input is ignored until the specified
753 .Sq Pf . Ar end
754 macro is encountered.
755 Do not use the escape character
756 .Sq \e
757 anywhere in the definition of
758 .Ar end ;
759 it would cause very strange behaviour.
760 .Pp
761 When the
762 .Ar end
763 macro is a roff request or a roff macro, like in
764 .Pp
765 .Dl \&.ig if
766 .Pp
767 the subsequent invocation of
768 .Sx \&if
769 will first terminate the
770 .Ar ignored text ,
771 then be invoked as usual.
772 Otherwise, it only terminates the
773 .Ar ignored text ,
774 and arguments following it or the
775 .Sq \&..
776 request are discarded.
777 .Ss \&ne
778 Declare the need for the specified minimum vertical space
779 before the next trap or the bottom of the page.
780 This line-scoped request is currently ignored.
781 .Ss \&nh
782 Turn off automatic hyphenation mode.

```

```

783 This line-scoped request is currently ignored.
784 .Ss \&rm
785 Remove a request, macro or string.
786 This request is intended to have one argument,
787 the name of the request, macro or string to be undefined.
788 Currently, it is ignored including its arguments,
789 and the number of arguments is not checked.
790 .Ss \&nr
791 Define a register.
792 A register is an arbitrary string value that defines some sort of state,
793 which influences parsing and/or formatting.
794 Its syntax is as follows:
795 .Pp
796 .Dl Pf \. Cm \&nr Ar name Ar value
797 .Pp
798 The
799 .Ar value
800 may, at the moment, only be an integer.
801 So far, only the following register
802 .Ar name
803 is recognised:
804 .Bl -tag -width Ds
805 .It Cm nS
806 If set to a positive integer value, certain
807 .Xr mdoc 5
808 macros will behave in the same way as in the
809 .Em SYNOPSIS
810 section.
811 If set to 0, these macros will behave in the same way as outside the
812 .Em SYNOPSIS
813 section, even when called within the
814 .Em SYNOPSIS
815 section itself.
816 Note that starting a new
817 .Xr mdoc 5
818 section with the
819 .Cm \&Sh
820 macro will reset this register.
821 .El
822 .Ss \&ns
823 Turn on no-space mode.
824 This line-scoped request is intended to take no arguments.
825 Currently, it is ignored including its arguments,
826 and the number of arguments is not checked.
827 .Ss \&ps
828 Change point size.
829 This line-scoped request is intended to take one numerical argument.
830 Currently, it is ignored including its arguments,
831 and the number of arguments is not checked.
832 .Ss \&so
833 Include a source file.
834 Its syntax is as follows:
835 .Pp
836 .Dl Pf \. Cm \&so Ar file
837 .Pp
838 The
839 .Ar file
840 will be read and its contents processed as input in place of the
841 .Sq \&.so
842 request line.
843 To avoid inadvertent inclusion of unrelated files,
844 .Xr mandoc 1
845 only accepts relative paths not containing the strings
846 .Qq ../
847 and
848 .Qq /.. .

```

```

849 .Pp
850 This request requires
851 .Xr man 1
852 to change to the right directory before calling
853 .Xr mandoc 1 ,
854 per convention to the root of the manual tree.
855 Typical usage looks like:
856 .Pp
857 .Dl \&.so man3/Xcursor.3
858 .Pp
859 As the whole concept is rather fragile, the use of
860 .Sx \&so
861 is discouraged.
862 Use
863 .Xr ln 1
864 instead.
865 .Ss \&ta
866 Set tab stops.
867 This line-scoped request can take an arbitrary number of arguments.
868 Currently, it is ignored including its arguments.
869 .Ss \&tr
870 Output character translation.
871 Its syntax is as follows:
872 .Pp
873 .Dl Pf \. Cm \&tr Ar [ab]+
874 .Pp
875 Pairs of
876 .Ar ab
877 characters are replaced
878 .Ar ( a
879 for
880 .Ar b ) .
881 Replacement (or origin) characters may also be character escapes; thus,
882 .Pp
883 .Dl tr \e(xx)\e(yy)
884 .Pp
885 replaces all invocations of \e(xx with \e(yy.
886 .Ss \&T&
887 Re-start a table layout, retaining the options of the prior table
888 invocation.
889 See
890 .Sx \&TS .
891 .Ss \&TE
892 End a table context.
893 See
894 .Sx \&TS .
895 .Ss \&TS
896 Begin a table, which formats input in aligned rows and columns.
897 See
898 .Xr tbl 5
899 for a description of the tbl language.
900 .Sh COMPATIBILITY
901 This section documents compatibility between mandoc and other other
902 .Nm
903 implementations, at this time limited to GNU troff
904 .Pq Qq groff .
905 The term
906 .Qq historic groff
907 refers to groff version 1.15.
908 .Pp
909 .Bl -dash -compact
910 .It
911 In mandoc, the
912 .Sx \&EQ ,
913 .Sx \&TE ,
914 .Sx \&TS ,

```

```

915 and
916 .Sx \&T& ,
917 macros are considered regular macros.
918 In all other
919 .Nm
920 implementations, these are special macros that must be specified without
921 spacing between the control character (which must be a period) and the
922 macro name.
923 .It
924 The
925 .Cm nS
926 register is only compatible with OpenBSD's groff-1.15.
927 .It
928 Historic groff did not accept white-space before a custom
929 .Ar end
930 macro for the
931 .Sx \&ig
932 request.
933 .It
934 The
935 .Sx \&if
936 and family would print funny white-spaces with historic groff when
937 using the next-line syntax.
938 .El
939 .Sh SEE ALSO
940 .Xr mandoc 1 ,
941 .Xr eqn 5 ,
942 .Xr man 5 ,
943 .Xr mandoc_char 5 ,
944 .Xr mdoc 5 ,
945 .Xr tbl 5
946 .Rs
947 .%A Joseph F. Ossanna
948 .%A Brian W. Kernighan
949 .%I AT&T Bell Laboratories
950 .%T Troff User's Manual
951 .%R Computing Science Technical Report
952 .%N 54
953 .%C Murray Hill, New Jersey
954 .%D 1976 and 1992
955 .%U http://www.kohala.com/start/troff/cstr54.ps
956 .Re
957 .Rs
958 .%A Joseph F. Ossanna
959 .%A Brian W. Kernighan
960 .%A Gunnar Ritter
961 .%T Heirloom Documentation Tools Nroff/Troff User's Manual
962 .%D September 17, 2007
963 .%U http://heirloom.sourceforge.net/doctools/troff.pdf
964 .Re
965 .Sh HISTORY
966 The RUNOFF typesetting system, whose input forms the basis for
967 .Nm ,
968 was written in MAD and FAP for the CTSS operating system by Jerome E.
969 Saltzer in 1964.
970 Doug McIlroy rewrote it in BCPL in 1969, renaming it
971 .Nm .
972 Dennis M. Ritchie rewrote McIlroy's
973 .Nm
974 in PDP-11 assembly for
975 .At v1 ,
976 Joseph F. Ossanna improved roff and renamed it nroff
977 for
978 .At v2 ,
979 then ported nroff to C as troff, which Brian W. Kernighan released with
980 .At v7 .

```

```

981 In 1989, James Clarke re-implemented troff in C++, naming it groff.
982 .Sh AUTHORS
983 .An -nosplit
984 This
985 .Nm
986 reference was written by
987 .An Kristaps Dzonsons ,
988 .Mt kristaps@bsd.lv ;
989 and
990 .An Ingo Schwarze ,
991 .Mt schwarze@openbsd.org .

```

```
*****
75905 Sat Jul 19 19:24:38 2014
new/usr/src/man/man5/mdoc.5
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
```

```
1 .\"
2 .\" Permission to use, copy, modify, and distribute this software for any
3 .\" purpose with or without fee is hereby granted, provided that the above
4 .\" copyright notice and this permission notice appear in all copies.
5 .\"
6 .\" THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
7 .\" WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
8 .\" MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
9 .\" ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
10 .\" WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
11 .\" ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
12 .\" OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
13 .\"
14 .\"
15 .\" Copyright (c) 2009, 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
16 .\" Copyright (c) 2010, 2011 Ingo Schwarze <schwarze@openbsd.org>
17 .\" Copyright 2012 Nexenta Systems, Inc. All rights reserved.
18 .\" Copyright 2014 Garrett D'Amore <garrett@dmaore.org>
19 .\"
20 .Dd Jul 19, 2014
21 .Dt MDOC 5
22 .Os
23 .Sh NAME
24 .Nm mdoc
25 .Nd semantic markup language for formatting manual pages
26 .Sh DESCRIPTION
27 The
28 .Nm mdoc
29 language supports authoring of manual pages for the
30 .Xr man 1
31 utility by allowing semantic annotations of words, phrases,
32 page sections and complete manual pages.
33 Such annotations are used by formatting tools to achieve a uniform
34 presentation across all manuals written in
35 .Nm ,
36 and to support hyperlinking if supported by the output medium.
37 .Pp
38 This reference document describes the structure of manual pages
39 and the syntax and usage of the
40 .Nm
41 language.
42 The reference implementation of a parsing and formatting tool is
43 .Xr mandoc 1 ;
44 the
45 .Sx COMPATIBILITY
46 section describes compatibility with other implementations.
47 .Pp
48 In an
49 .Nm
50 document, lines beginning with the control character
51 .Sq \&.
52 are called
53 .Dq macro lines .
54 The first word is the macro name.
55 It consists of two or three letters.
56 Most macro names begin with a capital letter.
```

```
57 For a list of available macros, see
58 .Sx MACRO OVERVIEW .
59 The words following the macro name are arguments to the macro, optionally
60 including the names of other, callable macros; see
61 .Sx MACRO SYNTAX
62 for details.
63 .Pp
64 Lines not beginning with the control character are called
65 .Dq text lines .
66 They provide free-form text to be printed; the formatting of the text
67 depends on the respective processing context:
68 .Bd -literal -offset indent
69 \&.Sh Macro lines change control state.
70 Text lines are interpreted within the current state.
71 .Ed
72 .Pp
73 Many aspects of the basic syntax of the
74 .Nm
75 language are based on the
76 .Xr roff 5
77 language; see the
78 .Em LANGUAGE SYNTAX
79 and
80 .Em MACRO SYNTAX
81 sections in the
82 .Xr roff 5
83 manual for details, in particular regarding
84 comments, escape sequences, whitespace, and quoting.
85 However, using
86 .Xr roff 5
87 requests in
88 .Nm
89 documents is discouraged;
90 .Xr mandoc 1
91 supports some of them merely for backward compatibility.
92 .Sh MANUAL STRUCTURE
93 A well-formed
94 .Nm
95 document consists of a document prologue followed by one or more
96 sections.
97 .Pp
98 The prologue, which consists of the
99 .Sx \&Dd ,
100 .Sx \&Dt ,
101 and
102 .Sx \&Os
103 macros in that order, is required for every document.
104 .Pp
105 The first section (sections are denoted by
106 .Sx \&Sh )
107 must be the NAME section, consisting of at least one
108 .Sx \&Nm
109 followed by
110 .Sx \&Nd .
111 .Pp
112 Following that, convention dictates specifying at least the
113 .Em SYNOPSIS
114 and
115 .Em DESCRIPTION
116 sections, although this varies between manual sections.
117 .Pp
118 The following is a well-formed skeleton
119 .Nm
120 file for a utility
121 .Qq progname :
122 .Bd -literal -offset indent
```

```

123 \&.Dd Jan 1, 1970
124 \&.Dt PROGNAME section
125 \&.Os
126 \&.Sh NAME
127 \&.Nm progname
128 \&.Nd one line description
129 \&.\e\(\dq .Sh LIBRARY
130 \&.\e\(\dq For sections 2, 3, & 9 only.
131 \&.Sh SYNOPSIS
132 \&.Nm progname
133 \&.Op Fl options
134 \&.Ar
135 \&.Sh DESCRIPTION
136 The
137 \&.Nm
138 utility processes files ...
139 \&.\e\(\dq .Sh IMPLEMENTATION NOTES
140 \&.\e\(\dq .Sh RETURN VALUES
141 \&.\e\(\dq For sections 2, 3, & 9 only.
142 \&.\e\(\dq .Sh ENVIRONMENT
143 \&.\e\(\dq For sections 1, 1M, and 5.
144 \&.\e\(\dq .Sh FILES
145 \&.\e\(\dq .Sh EXIT STATUS
146 \&.\e\(\dq For sections 1, 1M, and 5.
147 \&.\e\(\dq .Sh EXAMPLES
148 \&.\e\(\dq .Sh DIAGNOSTICS
149 \&.\e\(\dq .Sh ERRORS
150 \&.\e\(\dq For sections 2, 3, & 9 only.
151 \&.\e\(\dq .Sh ARCHITECTURE
152 \&.\e\(\dq .Sh CODE SET INDEPENDENCE
153 \&.\e\(\dq For sections 1, 1M, & 3 only.
154 \&.\e\(\dq .Sh INTERFACE STABILITY
155 \&.\e\(\dq .Sh MT-LEVEL
156 \&.\e\(\dq For sections 2 & 3 only.
157 \&.\e\(\dq .Sh SECURITY
158 \&.\e\(\dq .Sh SEE ALSO
159 \&.\e\(\dq .Xr foobar 1
160 \&.\e\(\dq .Sh STANDARDS
161 \&.\e\(\dq .Sh HISTORY
162 \&.\e\(\dq .Sh AUTHORS
163 \&.\e\(\dq .Sh CAVEATS
164 \&.\e\(\dq .Sh BUGS
165 .Ed
166 .Pp
167 The sections in an
168 .Nm
169 document are conventionally ordered as they appear above.
170 Sections should be composed as follows:
171 .Bl -ohang -offset Ds
172 .It Em NAME
173 The name(s) and a one line description of the documented material.
174 The syntax for this as follows:
175 .Bd -literal -offset indent
176 \&.Nm name0 ,
177 \&.Nm name1 ,
178 \&.Nm name2
179 \&.Nd a one line description
180 .Ed
181 .Pp
182 Multiple
183 .Sq \&.Nm
184 names should be separated by commas.
185 .Pp
186 The
187 .Sx \&.Nm
188 macro(s) must precede the

```

```

189 .Sx \&.Nd
190 macro.
191 .Pp
192 See
193 .Sx \&.Nm
194 and
195 .Sx \&.Nd .
196 .It Em LIBRARY
197 The name of the library containing the documented material, which is
198 assumed to be a function in a section 2, 3, or 9 manual.
199 The syntax for this is as follows:
200 .Bd -literal -offset indent
201 \&.Lb libarm
202 .Ed
203 .Pp
204 See
205 .Sx \&.Lb .
206 .It Em SYNOPSIS
207 Documents the utility invocation syntax, function call syntax, or device
208 configuration.
209 .Pp
210 For the first, utilities (sections 1 and 1M), this is
211 generally structured as follows:
212 .Bd -literal -offset indent
213 \&.Nm bar
214 \&.Op Fl v
215 \&.Op Fl o Ar file
216 \&.Op Ar
217 \&.Nm foo
218 \&.Op Fl v
219 \&.Op Fl o Ar file
220 \&.Op Ar
221 .Ed
222 .Pp
223 Commands should be ordered alphabetically.
224 .Pp
225 For the second, function calls (sections 2, 3, 7I, 7P, 9):
226 .Bd -literal -offset indent
227 \&.In header.h
228 \&.Vt extern const char *global;
229 \&.Ft "char *"
230 \&.Fn foo "const char *src"
231 \&.Ft "char *"
232 \&.Fn bar "const char *src"
233 .Ed
234 .Pp
235 Ordering of
236 .Sx \&.In ,
237 .Sx \&.Vt ,
238 .Sx \&.Fn ,
239 and
240 .Sx \&.Fo
241 macros should follow C header-file conventions.
242 .Pp
243 And for the third, configurations (section 7D):
244 .Bd -literal -offset indent
245 \&.Pa /dev/device_node
246 .Ed
247 .Pp
248 Manuals not in these sections generally don't need a
249 .Em SYNOPSIS .
250 .Pp
251 Some macros are displayed differently in the
252 .Em SYNOPSIS
253 section, particularly
254 .Sx \&.Nm ,

```

```

255 .Sx \&Cd ,
256 .Sx \&Fd ,
257 .Sx \&Fn ,
258 .Sx \&Fo ,
259 .Sx \&In ,
260 .Sx \&Vt ,
261 and
262 .Sx \&Ft .
263 All of these macros are output on their own line.
264 If two such dissimilar macros are pairwise invoked (except for
265 .Sx \&Ft
266 before
267 .Sx \&Fo
268 or
269 .Sx \&Fn ) ,
270 they are separated by a vertical space, unless in the case of
271 .Sx \&Fo ,
272 .Sx \&Fn ,
273 and
274 .Sx \&Ft ,
275 which are always separated by vertical space.
276 .Pp
277 When text and macros following an
278 .Sx \&Nm
279 macro starting an input line span multiple output lines,
280 all output lines but the first will be indented to align
281 with the text immediately following the
282 .Sx \&Nm
283 macro, up to the next
284 .Sx \&Nm ,
285 .Sx \&Sh ,
286 or
287 .Sx \&Ss
288 macro or the end of an enclosing block, whichever comes first.
289 .It Em DESCRIPTION
290 This begins with an expansion of the brief, one line description in
291 .Em NAME :
292 .Bd -literal -offset indent
293 The
294 \&.Nm
295 utility does this, that, and the other.
296 .Ed
297 .Pp
298 It usually follows with a breakdown of the options (if documenting a
299 command), such as:
300 .Bd -literal -offset indent
301 The arguments are as follows:
302 \&.Bl \-tag \-width Ds
303 \&.It Fl v
304 Print verbose information.
305 \&.El
306 .Ed
307 .Pp
308 Manuals not documenting a command won't include the above fragment.
309 .Pp
310 Since the
311 .Em DESCRIPTION
312 section usually contains most of the text of a manual, longer manuals
313 often use the
314 .Sx \&Ss
315 macro to form subsections.
316 In very long manuals, the
317 .Em DESCRIPTION
318 may be split into multiple sections, each started by an
319 .Sx \&Sh
320 macro followed by a non-standard section name, and each having

```

```

321 several subsections, like in the present
322 .Nm
323 manual.
324 .It Em IMPLEMENTATION NOTES
325 Implementation-specific notes should be kept here.
326 This is useful when implementing standard functions that may have side
327 effects or notable algorithmic implications.
328 .It Em RETURN VALUES
329 This section documents the
330 return values of functions in sections 2, 3, and 9.
331 .Pp
332 See
333 .Sx \&Rv .
334 .It Em ENVIRONMENT
335 Lists the environment variables used by the utility,
336 and explains the syntax and semantics of their values.
337 The
338 .Xr environ 5
339 manual provides examples of typical content and formatting.
340 .Pp
341 See
342 .Sx \&Ev .
343 .It Em FILES
344 Documents files used.
345 It's helpful to document both the file name and a short description of how
346 the file is used (created, modified, etc.).
347 .Pp
348 See
349 .Sx \&Pa .
350 .It Em EXIT STATUS
351 This section documents the
352 command exit status for sections 1 and 1M.
353 Historically, this information was described in
354 .Em DIAGNOSTICS ,
355 a practise that is now discouraged.
356 .Pp
357 See
358 .Sx \&Ex .
359 .It Em EXAMPLES
360 Example usages.
361 This often contains snippets of well-formed, well-tested invocations.
362 Make sure that examples work properly!
363 .It Em DIAGNOSTICS
364 Documents error and diagnostic messages displayed to the user or
365 sent to logs. Note that exit
366 status and return values should be documented in the
367 .Em EXIT STATUS
368 and
369 .Em RETURN VALUES
370 sections.
371 .Pp
372 See
373 .Sx \&Bl
374 .Fl diag .
375 .It Em ERRORS
376 Documents error handling in sections 2, 3, and 9.
377 .Pp
378 See
379 .Sx \&Er .
380 .It Em ARCHITECTURE
381 This section is usually absent, but will be present when the
382 interface is specific to one or more architectures.
383 .It Em CODE SET INDEPENDENCE
384 Indicates whether the interface operates correctly with various different
385 code sets. True independent code sets will support not only ASCII and
386 Extended UNIX Codesets (EUC), but also other multi-byte encodings such as

```


387 UTF-8 and GB2312.
 388 .Pp
 389 Generally there will be some limitations that are fairly standard. See
 390 .Xr standards 5 for more information about some of these. Most interfaces
 391 should support at least UTF-8 in addition to ASCII.
 392 .It Em INTERFACE STABILITY
 393 Indicates the level of commitment to the interface. Interfaces can be described
 394 with in the following ways:
 395 .Bl -tag -width Ds
 396 .It Nm Standard
 397 Indicates that the interface is defined by one or more standards bodies.
 398 Generally, changes to the interface will be carefully managed to conform
 399 to the relevant standards. These interfaces are generally the most suitable
 400 for use in portable programs.
 401 .It Nm Committed
 402 Indicates that the interface is intended to be preserved for the long-haul, and
 403 will rarely, if ever change, and never without notification (barring
 404 extraordinary and extenuating circumstances). These interfaces are
 405 preferred over other interfaces with the exception of
 406 .Nm Standard
 407 interfaces.
 408 .It Nm Uncommitted
 409 Indicates that the interface may change. Generally, changes to these interfaces
 410 should be infrequent, and some effort will be made to address compatibility
 411 considerations when changing or removing such interfaces. However, there is
 412 no firm commitment to the preservation of the interface. Most often this
 413 is applied to interfaces where operational experience with the interface
 414 is still limited and some need to change may be anticipated.
 415 .Pp
 416 Consumers should expect to revalidate any
 417 .Nm Uncommitted
 418 interfaces when crossing release boundaries. Products intended for
 419 use on many releases or intended to support compatibility with future
 420 releases should avoid these interfaces.
 421 .It Nm Volatile
 422 The interface can change at any time for any reason. Often this relates to
 423 interfaces that are part of external software components that are still evolving
 424 rapidly. Consumers should not expect that the interface (either binary or
 425 source level) will be unchanged from one release to the next.
 426 .It Nm Not-an-Interface
 427 Describes something that is specifically not intended for programmatic
 428 consumption. For example, specific human-readable output, or the layout
 429 of graphical items on a user interface, may be described this way. Generally
 430 programmatic alternatives to these will be available, and should be used
 431 when programmatic consumption is needed.
 432 .It Nm Private
 433 This is an internal interface. Generally these interfaces should only be
 434 used within the project, and should not be used by other programs or modules.
 435 The interface can and will change without notice as the project needs, at
 436 any time.
 437 .Pp
 438 Most often, Private interfaces will lack any documentation whatsoever, and
 439 generally any undocumented interface can be assumed to be Private.
 440 .It Nm Obsolete
 441 The interface is not intended for use in new projects or programs, and may
 442 be removed at a future date. The
 443 .Nm Obsolete
 444 word is a modifier that can
 445 be applied to other commitment levels. For example an
 446 .Nm Obsolete Committed
 447 interface is unlikely to be removed or changed, but nonetheless new use
 448 is discouraged (perhaps a better newer alternative is present).
 449 .El
 450 .It Em MT-LEVEL
 451 This section describes considerations for the interface when used within
 452 programs that use multiple threads. More discussion of these considerations

453 is made in the MT-Level section of
 454 .Xr attributes 5 .
 455 The interface can be described in the following ways.
 456 .Bl -tag -width Ds
 457 .It Nm Safe
 458 Indicates the interface is safe for use within multiple threads. There
 459 may be additional caveats that apply, in which case those will be
 460 described. Note that some interfaces have semantics which may affect
 461 other threads, but these should be an intrinsic part of the interface
 462 rather than an unexpected side effect. For example, closing a file in
 463 one thread will cause that file to be closed in all threads.
 464 .It Nm Unsafe
 465 Indicates the interface is unsuitable for concurrent use within multiple
 466 threads. A threaded application may still make use of the interface, but
 467 will be required to provide external synchronization means to ensure that
 468 only a single thread calls the interface at a time.
 469 .It Nm MT-Safe
 470 Indicates that the interface is not only safe for concurrent use, but is
 471 designed for such use. For example, a
 472 .Nm Safe
 473 interface may make use of a global lock to provide safety, but at reduced
 474 internal concurrency, whereas an
 475 .Nm MT-Safe
 476 interface will be designed to be efficient even when used concurrently.
 477 .It Nm Async-Signal-Safe
 478 Indicates that the library is safe for use within a signal handler. An
 479 .Nm MT-Safe
 480 interface can be made
 481 .Nm Async-Signal-Safe
 482 by ensuring that it blocks signals when acquiring locks.
 483 .It Nm Safe with Exections
 484 As for
 485 .Nm Safe
 486 but with specific exceptions noted.
 487 .It Nm MT-Safe with Exections
 488 As for
 489 .Nm MT-Safe
 490 but with specific exceptions noted.
 491 .El
 492 .It Em SECURITY
 493 Documents any security precautions that operators should consider.
 494 .It Em SEE ALSO
 495 References other manuals with related topics.
 496 This section should exist for most manuals.
 497 Cross-references should conventionally be ordered first by section, then
 498 alphabetically.
 499 .Pp
 500 References to other documentation concerning the topic of the manual page,
 501 for example authoritative books or journal articles, may also be
 502 provided in this section.
 503 .Pp
 504 See
 505 .Sx \&Rs
 506 and
 507 .Sx \&Xr .
 508 .It Em STANDARDS
 509 References any standards implemented or used.
 510 If not adhering to any standards, the
 511 .Em HISTORY
 512 section should be used instead.
 513 .Pp
 514 See
 515 .Sx \&St .
 516 .It Em HISTORY
 517 A brief history of the subject, including where it was first implemented,
 518 and when it was ported to or reimplemented for the operating system at hand.

```

519 .It Em AUTHORS
520 Credits to the person or persons who wrote the code and/or documentation.
521 Authors should generally be noted by both name and email address.
522 .Pp
523 See
524 .Sx \&An .
525 .It Em CAVEATS
526 Common misuses and misunderstandings should be explained
527 in this section.
528 .It Em BUGS
529 Known bugs, limitations, and work-arounds should be described
530 in this section.
531 .El
532 .Sh MACRO OVERVIEW
533 This overview is sorted such that macros of similar purpose are listed
534 together, to help find the best macro for any given purpose.
535 Deprecated macros are not included in the overview, but can be found below
536 in the alphabetical
537 .Sx MACRO REFERENCE .
538 .Ss Document preamble and NAME section macros
539 .Bl -column "Brq, Bro, Brc" description
540 .It Sx \&Dd Ta document date: Ar month day , year
541 .It Sx \&Dt Ta document title: Ar TITLE SECTION Op Ar volume | arch
542 .It Sx \&Os Ta operating system version: Op Ar system Op Ar version
543 .It Sx \&Nm Ta document name (one argument)
544 .It Sx \&Nd Ta document description (one line)
545 .El
546 .Ss Sections and cross references
547 .Bl -column "Brq, Bro, Brc" description
548 .It Sx \&Sh Ta section header (one line)
549 .It Sx \&Ss Ta subsection header (one line)
550 .It Sx \&Sx Ta internal cross reference to a section or subsection
551 .It Sx \&Xr Ta cross reference to another manual page: Ar name section
552 .It Sx \&Pp , \&Lp Ta start a text paragraph (no arguments)
553 .El
554 .Ss Displays and lists
555 .Bl -column "Brq, Bro, Brc" description
556 .It Sx \&Bd , \&Ed Ta display block:
557 .Fl Ar type
558 .Op Fl offset Ar width
559 .Op Fl compact
560 .It Sx \&Dl Ta indented display (one line)
561 .It Sx \&Dl Ta indented literal display (one line)
562 .It Sx \&Bl , \&El Ta list block:
563 .Fl Ar type
564 .Op Fl width Ar val
565 .Op Fl offset Ar val
566 .Op Fl compact
567 .It Sx \&It Ta list item (syntax depends on Fl Ar type )
568 .It Sx \&Ta Ta table cell separator in Sx \&Bl Fl column No lists
569 .It Sx \&Rs , \&%* , \&Re Ta bibliographic block (references)
570 .El
571 .Ss Spacing control
572 .Bl -column "Brq, Bro, Brc" description
573 .It Sx \&Pf Ta prefix, no following horizontal space (one argument)
574 .It Sx \&Ns Ta roman font, no preceding horizontal space (no arguments)
575 .It Sx \&Ap Ta apostrophe without surrounding whitespace (no arguments)
576 .It Sx \&Sm Ta switch horizontal spacing mode: Cm on | off
577 .It Sx \&Bk , \&Ek Ta keep block: Fl words
578 .It Sx \&Br Ta force output line break in text mode (no arguments)
579 .It Sx \&sp Ta force vertical space: Op Ar height
580 .El
581 .Ss Semantic markup for command line utilities:
582 .Bl -column "Brq, Bro, Brc" description
583 .It Sx \&Nm Ta start a SYNOPSIS block with the name of a utility
584 .It Sx \&Fl Ta command line options (flags) (>=0 arguments)

```

```

585 .It Sx \&Cm Ta command modifier (>0 arguments)
586 .It Sx \&Ar Ta command arguments (>=0 arguments)
587 .It Sx \&Op , \&Oo , \&Oc Ta optional syntax elements (enclosure)
588 .It Sx \&IC Ta internal or interactive command (>0 arguments)
589 .It Sx \&Ev Ta environmental variable (>0 arguments)
590 .It Sx \&Pa Ta file system path (>=0 arguments)
591 .El
592 .Ss Semantic markup for function libraries:
593 .Bl -column "Brq, Bro, Brc" description
594 .It Sx \&Lb Ta function library (one argument)
595 .It Sx \&In Ta include file (one argument)
596 .It Sx \&Ft Ta function type (>0 arguments)
597 .It Sx \&Fo , \&Fc Ta function block: Ar funcname
598 .It Sx \&Fn Ta function name:
599 .Op Ar functype
600 .Ar funcname
601 .Oo
602 .Op Ar argtype
603 .Ar argname
604 .Oc
605 .It Sx \&Fa Ta function argument (>0 arguments)
606 .It Sx \&Vt Ta variable type (>0 arguments)
607 .It Sx \&Va Ta variable name (>0 arguments)
608 .It Sx \&Dv Ta defined variable or preprocessor constant (>0 arguments)
609 .It Sx \&Er Ta error constant (>0 arguments)
610 .It Sx \&Ev Ta environmental variable (>0 arguments)
611 .El
612 .Ss Various semantic markup:
613 .Bl -column "Brq, Bro, Brc" description
614 .It Sx \&An Ta author name (>0 arguments)
615 .It Sx \&Lk Ta hyperlink: Ar uri Op Ar name
616 .It Sx \&Mt Ta Do mailto Dc hyperlink: Ar address
617 .It Sx \&Cd Ta kernel configuration declaration (>0 arguments)
618 .It Sx \&Ad Ta memory address (>0 arguments)
619 .It Sx \&Ms Ta mathematical symbol (>0 arguments)
620 .It Sx \&Tn Ta tradename (>0 arguments)
621 .El
622 .Ss Physical markup
623 .Bl -column "Brq, Bro, Brc" description
624 .It Sx \&Em Ta italic font or underline (emphasis) (>0 arguments)
625 .It Sx \&Sy Ta boldface font (symbolic) (>0 arguments)
626 .It Sx \&Li Ta typewriter font (literal) (>0 arguments)
627 .It Sx \&No Ta return to roman font (normal) (no arguments)
628 .It Sx \&Bf , \&Ef Ta font block:
629 .Op Fl Ar type | Cm \&Em | \&Li | \&Sy
630 .El
631 .Ss Physical enclosures
632 .Bl -column "Brq, Bro, Brc" description
633 .It Sx \&Dq , \&Do , \&Dc Ta enclose in typographic double quotes: Dq text
634 .It Sx \&Qq , \&Qo , \&Qc Ta enclose in typewriter double quotes: Qq text
635 .It Sx \&Sq , \&So , \&Sc Ta enclose in single quotes: Sq text
636 .It Sx \&Ql Ta single-quoted literal text: Ql text
637 .It Sx \&Pq , \&Po , \&Pc Ta enclose in parentheses: Pq text
638 .It Sx \&Bq , \&Bo , \&Bc Ta enclose in square brackets: Bq text
639 .It Sx \&Brq , \&Bro , \&Brc Ta enclose in curly braces: Brq text
640 .It Sx \&Aq , \&Ao , \&Ac Ta enclose in angle brackets: Aq text
641 .It Sx \&Eo , \&Ec Ta generic enclosure
642 .El
643 .Ss Text production
644 .Bl -column "Brq, Bro, Brc" description
645 .It Sx \&Ex Fl std Ta standard command exit values: Op Ar utility ...
646 .It Sx \&Rv Fl std Ta standard function return values: Op Ar function ...
647 .It Sx \&St Ta reference to a standards document (one argument)
648 .It Sx \&Ux Ta Ux
649 .It Sx \&At Ta At
650 .It Sx \&Bx Ta Bx

```

```

651 .It Sx \&Bsx Ta Bsx
652 .It Sx \&Nx Ta Nx
653 .It Sx \&Fx Ta Fx
654 .It Sx \&Ox Ta Ox
655 .It Sx \&Dx Ta Dx
656 .El
657 .Sh MACRO REFERENCE
658 This section is a canonical reference of all macros, arranged
659 alphabetically.
660 For the scoping of individual macros, see
661 .Sx MACRO SYNTAX .
662 .Ss \&%A
663 Author name of an
664 .Sx \&Rs
665 block.
666 Multiple authors should each be accorded their own
667 .Sx \&%A
668 line.
669 Author names should be ordered with full or abbreviated forename(s)
670 first, then full surname.
671 .Ss \&%B
672 Book title of an
673 .Sx \&Rs
674 block.
675 This macro may also be used in a non-bibliographic context when
676 referring to book titles.
677 .Ss \&%C
678 Publication city or location of an
679 .Sx \&Rs
680 block.
681 .Ss \&%D
682 Publication date of an
683 .Sx \&Rs
684 block.
685 Recommended formats of arguments are
686 .Ar month day , year
687 or just
688 .Ar year .
689 .Ss \&%I
690 Publisher or issuer name of an
691 .Sx \&Rs
692 block.
693 .Ss \&%J
694 Journal name of an
695 .Sx \&Rs
696 block.
697 .Ss \&%N
698 Issue number (usually for journals) of an
699 .Sx \&Rs
700 block.
701 .Ss \&%O
702 Optional information of an
703 .Sx \&Rs
704 block.
705 .Ss \&%P
706 Book or journal page number of an
707 .Sx \&Rs
708 block.
709 .Ss \&%Q
710 Institutional author (school, government, etc.) of an
711 .Sx \&Rs
712 block.
713 Multiple institutional authors should each be accorded their own
714 .Sx \&%Q
715 line.
716 .Ss \&%R

```

```

717 Technical report name of an
718 .Sx \&Rs
719 block.
720 .Ss \&%T
721 Article title of an
722 .Sx \&Rs
723 block.
724 This macro may also be used in a non-bibliographical context when
725 referring to article titles.
726 .Ss \&%U
727 URI of reference document.
728 .Ss \&%V
729 Volume number of an
730 .Sx \&Rs
731 block.
732 .Ss \&Ac
733 Close an
734 .Sx \&Ao
735 block.
736 Does not have any tail arguments.
737 .Ss \&Ad
738 Memory address.
739 Do not use this for postal addresses.
740 .Pp
741 Examples:
742 .Dl \&.Ad [0,$]
743 .Dl \&.Ad 0x00000000
744 .Ss \&An
745 Author name.
746 Can be used both for the authors of the program, function, or driver
747 documented in the manual, or for the authors of the manual itself.
748 Requires either the name of an author or one of the following arguments:
749 .Pp
750 .Bl -tag -width "-nosplitX" -offset indent -compact
751 .It Fl split
752 Start a new output line before each subsequent invocation of
753 .Sx \&An .
754 .It Fl nosplit
755 The opposite of
756 .Fl split .
757 .El
758 .Pp
759 The default is
760 .Fl nosplit .
761 The effect of selecting either of the
762 .Fl split
763 modes ends at the beginning of the
764 .Em AUTHORS
765 section.
766 In the
767 .Em AUTHORS
768 section, the default is
769 .Fl nosplit
770 for the first author listing and
771 .Fl split
772 for all other author listings.
773 .Pp
774 Examples:
775 .Dl \&.An -nosplit
776 .Dl \&.An Kristaps Dzonsons \&Aq kristaps@bsd.lv
777 .Ss \&Ao
778 Begin a block enclosed by angle brackets.
779 Does not have any head arguments.
780 .Pp
781 Examples:
782 .Dl \&.Fl -key= \&Ns \&Ao \&Ar val \&Ac

```

```

783 .Pp
784 See also
785 .Sx \&Aq .
786 .Ss \&Ap
787 Inserts an apostrophe without any surrounding whitespace.
788 This is generally used as a grammatical device when referring to the verb
789 form of a function.
790 .Pp
791 Examples:
792 .Dl \&.Fn execve \&Ap d
793 .Ss \&Aq
794 Encloses its arguments in angle brackets.
795 .Pp
796 Examples:
797 .Dl \&.Fl -key= \&Ns \&Aq \&Ar val
798 .Pp
799 .Em Remarks :
800 this macro is often abused for rendering URIs, which should instead use
801 .Sx \&Lk
802 or
803 .Sx \&Mt ,
804 or to note pre-processor
805 .Dq Li #include
806 statements, which should use
807 .Sx \&In .
808 .Pp
809 See also
810 .Sx \&Ao .
811 .Ss \&Ar
812 Command arguments.
813 If an argument is not provided, the string
814 .Dq file ...\&
815 is used as a default.
816 .Pp
817 Examples:
818 .Dl ".Fl o Ar file"
819 .Dl ".Ar"
820 .Dl ".Ar arg1 , arg2 ."
821 .Pp
822 The arguments to the
823 .Sx \&Ar
824 macro are names and placeholders for command arguments;
825 for fixed strings to be passed verbatim as arguments, use
826 .Sx \&Fl
827 or
828 .Sx \&Cm .
829 .Ss \&At
830 Formats an AT&T version.
831 Accepts one optional argument:
832 .Pp
833 .Bl -tag -width "v[1-7] | 32vX" -offset indent -compact
834 .It Cm v[1-7] | 32v
835 A version of
836 .At .
837 .It Cm III
838 .At III .
839 .It Cm V[. [1-4]]?
840 A version of
841 .At V .
842 .El
843 .Pp
844 Note that these arguments do not begin with a hyphen.
845 .Pp
846 Examples:
847 .Dl \&.At
848 .Dl \&.At III

```

```

849 .Dl \&.At V.1
850 .Pp
851 See also
852 .Sx \&BSx ,
853 .Sx \&Bx ,
854 .Sx \&Dx ,
855 .Sx \&Fx ,
856 .Sx \&Nx ,
857 .Sx \&Ox ,
858 and
859 .Sx \&Ux .
860 .Ss \&Bc
861 Close a
862 .Sx \&Bo
863 block.
864 Does not have any tail arguments.
865 .Ss \&Bd
866 Begin a display block.
867 Its syntax is as follows:
868 .Bd -ragged -offset indent
869 .Pf \. Sx \&Bd
870 .Fl Ns Ar type
871 .Op Fl offset Ar width
872 .Op Fl compact
873 .Ed
874 .Pp
875 Display blocks are used to select a different indentation and
876 justification than the one used by the surrounding text.
877 They may contain both macro lines and text lines.
878 By default, a display block is preceded by a vertical space.
879 .Pp
880 The
881 .Ar type
882 must be one of the following:
883 .Bl -tag -width l3n -offset indent
884 .It Fl centered
885 Produce one output line from each input line, and centre-justify each line.
886 Using this display type is not recommended; many
887 .Nm
888 implementations render it poorly.
889 .It Fl filled
890 Change the positions of line breaks to fill each line, and left- and
891 right-justify the resulting block.
892 .It Fl literal
893 Produce one output line from each input line,
894 and do not justify the block at all.
895 Preserve white space as it appears in the input.
896 Always use a constant-width font.
897 Use this for displaying source code.
898 .It Fl ragged
899 Change the positions of line breaks to fill each line, and left-justify
900 the resulting block.
901 .It Fl unfilled
902 The same as
903 .Fl literal ,
904 but using the same font as for normal text, which is a variable width font
905 if supported by the output device.
906 .El
907 .Pp
908 The
909 .Ar type
910 must be provided first.
911 Additional arguments may follow:
912 .Bl -tag -width l3n -offset indent
913 .It Fl offset Ar width
914 Indent the display by the

```

```

915 .Ar width ,
916 which may be one of the following:
917 .Bl -item
918 .It
919 One of the pre-defined strings
920 .Cm indent ,
921 the width of a standard indentation (six constant width characters);
922 .Cm indent-two ,
923 twice
924 .Cm indent ;
925 .Cm left ,
926 which has no effect;
927 .Cm right ,
928 which justifies to the right margin; or
929 .Cm center ,
930 which aligns around an imagined centre axis.
931 .It
932 A macro invocation, which selects a predefined width
933 associated with that macro.
934 The most popular is the imaginary macro
935 .Ar \&Ds ,
936 which resolves to
937 .Sy 6n .
938 .It
939 A width using the syntax described in
940 .Sx Scaling Widths .
941 .It
942 An arbitrary string, which indents by the length of this string.
943 .El
944 .Pp
945 When the argument is missing,
946 .Fl offset
947 is ignored.
948 .It Fl compact
949 Do not assert vertical space before the display.
950 .El
951 .Pp
952 Examples:
953 .Bd -literal -offset indent
954 \&.Bd \-literal \-offset indent \-compact
955 Hello world.
956 \&.Ed
957 .Ed
958 .Pp
959 See also
960 .Sx \&Dl
961 and
962 .Sx \&Dl .
963 .Ss \&Bf
964 Change the font mode for a scoped block of text.
965 Its syntax is as follows:
966 .Bd -ragged -offset indent
967 .Pf \. Sx \&Bf
968 .Oo
969 .Fl emphasis | literal | symbolic |
970 .Cm \&Em | \&Li | \&Sy
971 .Oc
972 .Ed
973 .Pp
974 The
975 .Fl emphasis
976 and
977 .Cm \&Em
978 argument are equivalent, as are
979 .Fl symbolic
980 and

```

```

981 .Cm \&Sy ,
982 and
983 .Fl literal
984 and
985 .Cm \&Li .
986 Without an argument, this macro does nothing.
987 The font mode continues until broken by a new font mode in a nested
988 scope or
989 .Sx \&Ef
990 is encountered.
991 .Pp
992 See also
993 .Sx \&Li ,
994 .Sx \&Ef ,
995 .Sx \&Em ,
996 and
997 .Sx \&Sy .
998 .Ss \&Bk
999 For each macro, keep its output together on the same output line,
1000 until the end of the macro or the end of the input line is reached,
1001 whichever comes first.
1002 Line breaks in text lines are unaffected.
1003 The syntax is as follows:
1004 .Pp
1005 .Dl Pf \. Sx \&Bk Fl words
1006 .Pp
1007 The
1008 .Fl words
1009 argument is required; additional arguments are ignored.
1010 .Pp
1011 The following example will not break within each
1012 .Sx \&Op
1013 macro line:
1014 .Bd -literal -offset indent
1015 \&.Bk \-words
1016 \&.Op Fl f Ar flags
1017 \&.Op Fl o Ar output
1018 \&.Ek
1019 .Ed
1020 .Pp
1021 Be careful in using over-long lines within a keep block!
1022 Doing so will clobber the right margin.
1023 .Ss \&Bl
1024 Begin a list.
1025 Lists consist of items specified using the
1026 .Sx \&It
1027 macro, containing a head or a body or both.
1028 The list syntax is as follows:
1029 .Bd -ragged -offset indent
1030 .Pf \. Sx \&Bl
1031 .Fl Ns Ar type
1032 .Op Fl width Ar val
1033 .Op Fl offset Ar val
1034 .Op Fl compact
1035 .Op HEAD ...
1036 .Ed
1037 .Pp
1038 The list
1039 .Ar type
1040 is mandatory and must be specified first.
1041 The
1042 .Fl width
1043 and
1044 .Fl offset
1045 arguments accept
1046 .Sx Scaling Widths

```

1047 or use the length of the given string.
 1048 The
 1049 .Fl offset
 1050 is a global indentation for the whole list, affecting both item heads
 1051 and bodies.
 1052 For those list types supporting it, the
 1053 .Fl width
 1054 argument requests an additional indentation of item bodies,
 1055 to be added to the
 1056 .Fl offset .
 1057 Unless the
 1058 .Fl compact
 1059 argument is specified, list entries are separated by vertical space.
 1060 .Pp
 1061 A list must specify one of the following list types:
 1062 .Bl -tag -width l2n -offset indent
 1063 .It Fl bullet
 1064 No item heads can be specified, but a bullet will be printed at the head
 1065 of each item.
 1066 Item bodies start on the same output line as the bullet
 1067 and are indented according to the
 1068 .Fl width
 1069 argument.
 1070 .It Fl column
 1071 A columnated list.
 1072 The
 1073 .Fl width
 1074 argument has no effect; instead, each argument specifies the width
 1075 of one column, using either the
 1076 .Sx Scaling Widths
 1077 syntax or the string length of the argument.
 1078 If the first line of the body of a
 1079 .Fl column
 1080 list is not an
 1081 .Sx \&It
 1082 macro line,
 1083 .Sx \&It
 1084 contexts spanning one input line each are implied until an
 1085 .Sx \&It
 1086 macro line is encountered, at which point items start being interpreted as
 1087 described in the
 1088 .Sx \&It
 1089 documentation.
 1090 .It Fl dash
 1091 Like
 1092 .Fl bullet ,
 1093 except that dashes are used in place of bullets.
 1094 .It Fl diag
 1095 Like
 1096 .Fl inset ,
 1097 except that item heads are not parsed for macro invocations.
 1098 Most often used in the
 1099 .Em DIAGNOSTICS
 1100 section with error constants in the item heads.
 1101 .It Fl enum
 1102 A numbered list.
 1103 No item heads can be specified.
 1104 Formatted like
 1105 .Fl bullet ,
 1106 except that cardinal numbers are used in place of bullets,
 1107 starting at 1.
 1108 .It Fl hang
 1109 Like
 1110 .Fl tag ,
 1111 except that the first lines of item bodies are not indented, but follow
 1112 the item heads like in

1113 .Fl inset
 1114 lists.
 1115 .It Fl hyphen
 1116 Synonym for
 1117 .Fl dash .
 1118 .It Fl inset
 1119 Item bodies follow items heads on the same line, using normal inter-word
 1120 spacing.
 1121 Bodies are not indented, and the
 1122 .Fl width
 1123 argument is ignored.
 1124 .It Fl item
 1125 No item heads can be specified, and none are printed.
 1126 Bodies are not indented, and the
 1127 .Fl width
 1128 argument is ignored.
 1129 .It Fl chang
 1130 Item bodies start on the line following item heads and are not indented.
 1131 The
 1132 .Fl width
 1133 argument is ignored.
 1134 .It Fl tag
 1135 Item bodies are indented according to the
 1136 .Fl width
 1137 argument.
 1138 When an item head fits inside the indentation, the item body follows
 1139 this head on the same output line.
 1140 Otherwise, the body starts on the output line following the head.
 1141 .El
 1142 .Pp
 1143 Lists may be nested within lists and displays.
 1144 Nesting of
 1145 .Fl column
 1146 and
 1147 .Fl enum
 1148 lists may not be portable.
 1149 .Pp
 1150 See also
 1151 .Sx \&El
 1152 and
 1153 .Sx \&It .
 1154 .Ss \&Bo
 1155 Begin a block enclosed by square brackets.
 1156 Does not have any head arguments.
 1157 .Pp
 1158 Examples:
 1159 .Bd -literal -offset indent -compact
 1160 \&.Bo 1 ,
 1161 \&.Dv BUFSIZ \&Bc
 1162 .Ed
 1163 .Pp
 1164 See also
 1165 .Sx \&Bq .
 1166 .Ss \&Bq
 1167 Encloses its arguments in square brackets.
 1168 .Pp
 1169 Examples:
 1170 .Dl \&.Bq 1 , \&Dv BUFSIZ
 1171 .Pp
 1172 .Em Remarks :
 1173 this macro is sometimes abused to emulate optional arguments for
 1174 commands; the correct macros to use for this purpose are
 1175 .Sx \&Op ,
 1176 .Sx \&Oo ,
 1177 and
 1178 .Sx \&Oc .

```

1179 .Pp
1180 See also
1181 .Sx \&Bo .
1182 .Ss \&Brc
1183 Close a
1184 .Sx \&Bro
1185 block.
1186 Does not have any tail arguments.
1187 .Ss \&Bro
1188 Begin a block enclosed by curly braces.
1189 Does not have any head arguments.
1190 .Pp
1191 Examples:
1192 .Bd -literal -offset indent -compact
1193 \&.Bro 1 , ... ,
1194 \&.Va n \&Brc
1195 .Ed
1196 .Pp
1197 See also
1198 .Sx \&Brq .
1199 .Ss \&Brq
1200 Encloses its arguments in curly braces.
1201 .Pp
1202 Examples:
1203 .Dl \&.Brq 1 , ... , \&Va n
1204 .Pp
1205 See also
1206 .Sx \&Bro .
1207 .Ss \&BSx
1208 Format the BSD/OS version provided as an argument, or a default value if
1209 no argument is provided.
1210 .Pp
1211 Examples:
1212 .Dl \&.BSx 1.0
1213 .Dl \&.BSx
1214 .Pp
1215 See also
1216 .Sx \&At ,
1217 .Sx \&Bx ,
1218 .Sx \&Dx ,
1219 .Sx \&Fx ,
1220 .Sx \&Nx ,
1221 .Sx \&Ox ,
1222 and
1223 .Sx \&Ux .
1224 .Ss \&Bt
1225 Prints
1226 .Dq is currently in beta test.
1227 .Ss \&Bx
1228 Format the BSD version provided as an argument, or a default value if no
1229 argument is provided.
1230 .Pp
1231 Examples:
1232 .Dl \&.Bx 4.3 Tahoe
1233 .Dl \&.Bx 4.4
1234 .Dl \&.Bx
1235 .Pp
1236 See also
1237 .Sx \&At ,
1238 .Sx \&BSx ,
1239 .Sx \&Dx ,
1240 .Sx \&Fx ,
1241 .Sx \&Nx ,
1242 .Sx \&Ox ,
1243 and
1244 .Sx \&Ux .

```

```

1245 .Ss \&Cd
1246 Kernel configuration declaration. It is found in pages for
1247 .Bx
1248 and not used here.
1249 .Pp
1250 Examples:
1251 .Dl \&.Cd device le0 at scode?
1252 .Pp
1253 .Em Remarks :
1254 this macro is commonly abused by using quoted literals to retain
1255 whitespace and align consecutive
1256 .Sx \&Cd
1257 declarations.
1258 This practise is discouraged.
1259 .Ss \&Cm
1260 Command modifiers.
1261 Typically used for fixed strings passed as arguments, unless
1262 .Sx \&Fl
1263 is more appropriate.
1264 Also useful when specifying configuration options or keys.
1265 .Pp
1266 Examples:
1267 .Dl ".Nm mt Fl f Ar device Cm rewind"
1268 .Dl ".Nm ps Fl o Cm pid , Ns Cm command"
1269 .Dl ".Nm dd Cm if= Ns Ar file1 Cm of= Ns Ar file2"
1270 .Dl ".Cm IdentityFile Pa ~/.ssh/id_rsa"
1271 .Dl ".Cm LogLevel Dv DEBUG"
1272 .Ss \&Dl
1273 One-line indented display.
1274 This is formatted by the default rules and is useful for simple indented
1275 statements.
1276 It is followed by a newline.
1277 .Pp
1278 Examples:
1279 .Dl \&.Dl \&Fl abcdefgh
1280 .Pp
1281 See also
1282 .Sx \&Bd
1283 and
1284 .Sx \&Dl .
1285 .Ss \&Db
1286 Switch debugging mode.
1287 Its syntax is as follows:
1288 .Pp
1289 .Dl Pf \. Sx \&Db Cm on | off
1290 .Pp
1291 This macro is ignored by
1292 .Xr mandoc 1 .
1293 .Ss \&Dc
1294 Close a
1295 .Sx \&Do
1296 block.
1297 Does not have any tail arguments.
1298 .Ss \&Dd
1299 Document date.
1300 This is the mandatory first macro of any
1301 .Nm
1302 manual.
1303 Its syntax is as follows:
1304 .Pp
1305 .Dl Pf \. Sx \&Dd Ar month day , year
1306 .Pp
1307 The
1308 .Ar month
1309 is the full English month name, the
1310 .Ar day

```

1311 is an optionally zero-padded numeral, and the
 1312 .Ar year
 1313 is the full four-digit year.
 1314 .Pp
 1315 Other arguments are not portable; the
 1316 .Xr mandoc 1
 1317 utility handles them as follows:
 1318 .Bl -dash -offset 3n -compact
 1319 .It
 1320 To have the date automatically filled in by the
 1321 .Ox
 1322 version of
 1323 .Xr cvs 1 ,
 1324 the special string
 1325 .Dq \$&Mdocdate\$
 1326 can be given as an argument.
 1327 .It
 1328 A few alternative date formats are accepted as well
 1329 and converted to the standard form.
 1330 .It
 1331 If a date string cannot be parsed, it is used verbatim.
 1332 .It
 1333 If no date string is given, the current date is used.
 1334 .El
 1335 .Pp
 1336 Examples:
 1337 .Dl \&.Dd \$&Mdocdate\$
 1338 .Dl \&.Dd \$&Mdocdate: July 21 2007\$
 1339 .Dl \&.Dd July 21, 2007
 1340 .Pp
 1341 See also
 1342 .Sx \&Dt
 1343 and
 1344 .Sx \&Os .
 1345 .Ss \&Dl
 1346 One-line intended display.
 1347 This is formatted as literal text and is useful for commands and
 1348 invocations.
 1349 It is followed by a newline.
 1350 .Pp
 1351 Examples:
 1352 .Dl \&.Dl % mandoc mdoc.5 \e(ba less
 1353 .Pp
 1354 See also
 1355 .Sx \&Bd
 1356 and
 1357 .Sx \&Dl .
 1358 .Ss \&Do
 1359 Begin a block enclosed by double quotes.
 1360 Does not have any head arguments.
 1361 .Pp
 1362 Examples:
 1363 .Bd -literal -offset indent -compact
 1364 \&.Do
 1365 April is the cruellest month
 1366 \&.Dc
 1367 \e(em T.S. Eliot
 1368 .Ed
 1369 .Pp
 1370 See also
 1371 .Sx \&Dq .
 1372 .Ss \&Dq
 1373 Encloses its arguments in
 1374 .Dq typographic
 1375 double-quotes.
 1376 .Pp

1377 Examples:
 1378 .Bd -literal -offset indent -compact
 1379 \&.Dq April is the cruellest month
 1380 \e(em T.S. Eliot
 1381 .Ed
 1382 .Pp
 1383 See also
 1384 .Sx \&Qq ,
 1385 .Sx \&Sq ,
 1386 and
 1387 .Sx \&Do .
 1388 .Ss \&Dt
 1389 Document title.
 1390 This is the mandatory second macro of any
 1391 .Nm
 1392 file.
 1393 Its syntax is as follows:
 1394 .Bd -ragged -offset indent
 1395 .Pf \. Sx \&Dt
 1396 .Oo
 1397 .Ar title
 1398 .Oo
 1399 .Ar section
 1400 .Op Ar volume
 1401 .Op Ar arch
 1402 .Oc
 1403 .Oc
 1404 .Ed
 1405 .Pp
 1406 Its arguments are as follows:
 1407 .Bl -tag -width Ds -offset Ds
 1408 .It Ar title
 1409 The document's title (name), defaulting to
 1410 .Dq UNKNOWN
 1411 if unspecified.
 1412 It should be capitalised.
 1413 .It Ar section
 1414 The manual section. It should correspond to the manual's filename suffix
 1415 and defaults to
 1416 .Dq 1
 1417 if unspecified.
 1418 .It Ar volume
 1419 This overrides the volume inferred from
 1420 .Ar section .
 1421 This field is optional.
 1422 .It Ar arch
 1423 This specifies the machine architecture a manual page applies to,
 1424 where relevant.
 1425 .El
 1426 .Ss \&Dv
 1427 Defined variables such as preprocessor constants, constant symbols,
 1428 enumeration values, and so on.
 1429 .Pp
 1430 Examples:
 1431 .Dl \&.Dv NULL
 1432 .Dl \&.Dv BUFSIZ
 1433 .Dl \&.Dv STDOUT_FILENO
 1434 .Pp
 1435 See also
 1436 .Sx \&Er
 1437 and
 1438 .Sx \&Ev
 1439 for special-purpose constants and
 1440 .Sx \&Va
 1441 for variable symbols.
 1442 .Ss \&Dx


```

1443 Format the DragonFly BSD version provided as an argument, or a default
1444 value if no argument is provided.
1445 .Pp
1446 Examples:
1447 .Dl \&.Dx 2.4.1
1448 .Dl \&.Dx
1449 .Pp
1450 See also
1451 .Sx \&At ,
1452 .Sx \&Bsx ,
1453 .Sx \&Bx ,
1454 .Sx \&Fx ,
1455 .Sx \&Nx ,
1456 .Sx \&Ox ,
1457 and
1458 .Sx \&Ux .
1459 .Ss \&Ec
1460 Close a scope started by
1461 .Sx \&Eo .
1462 Its syntax is as follows:
1463 .Pp
1464 .Dl Pf \. Sx \&Ec Op Ar TERM
1465 .Pp
1466 The
1467 .Ar TERM
1468 argument is used as the enclosure tail, for example, specifying \e(rq
1469 will emulate
1470 .Sx \&Dc .
1471 .Ss \&Ed
1472 End a display context started by
1473 .Sx \&Bd .
1474 .Ss \&Ef
1475 End a font mode context started by
1476 .Sx \&Bf .
1477 .Ss \&Ek
1478 End a keep context started by
1479 .Sx \&Bk .
1480 .Ss \&El
1481 End a list context started by
1482 .Sx \&Bl .
1483 .Pp
1484 See also
1485 .Sx \&Bl
1486 and
1487 .Sx \&It .
1488 .Ss \&Em
1489 Denotes text that should be
1490 .Em emphasised .
1491 Note that this is a presentation term and should not be used for
1492 stylistically decorating technical terms.
1493 Depending on the output device, this is usually represented
1494 using an italic font or underlined characters.
1495 .Pp
1496 Examples:
1497 .Dl \&.Em Warnings!
1498 .Dl \&.Em Remarks :
1499 .Pp
1500 See also
1501 .Sx \&Bf ,
1502 .Sx \&Li ,
1503 .Sx \&No ,
1504 and
1505 .Sx \&Sy .
1506 .Ss \&En
1507 This macro is obsolete and not implemented in
1508 .Xr mandoc 1 .

```

```

1509 .Ss \&Eo
1510 An arbitrary enclosure.
1511 Its syntax is as follows:
1512 .Pp
1513 .Dl Pf \. Sx \&Eo Op Ar TERM
1514 .Pp
1515 The
1516 .Ar TERM
1517 argument is used as the enclosure head, for example, specifying \e(lq
1518 will emulate
1519 .Sx \&Do .
1520 .Ss \&Er
1521 Error constants for definitions of the
1522 .Va errno
1523 libc global variable.
1524 This is most often used in section 2 and 3 manual pages.
1525 .Pp
1526 Examples:
1527 .Dl \&.Er EPERM
1528 .Dl \&.Er ENOENT
1529 .Pp
1530 See also
1531 .Sx \&Dv
1532 for general constants.
1533 .Ss \&Es
1534 This macro is obsolete and not implemented.
1535 .Ss \&Ev
1536 Environmental variables such as those specified in
1537 .Xr environ 5 .
1538 .Pp
1539 Examples:
1540 .Dl \&.Ev DISPLAY
1541 .Dl \&.Ev PATH
1542 .Pp
1543 See also
1544 .Sx \&Dv
1545 for general constants.
1546 .Ss \&Ex
1547 Insert a standard sentence regarding command exit values of 0 on success
1548 and >0 on failure.
1549 This is most often used in section 1 and 1M manual pages.
1550 Its syntax is as follows:
1551 .Pp
1552 .Dl Pf \. Sx \&Ex Fl std Op Ar utility ...
1553 .Pp
1554 If
1555 .Ar utility
1556 is not specified, the document's name set by
1557 .Sx \&Nm
1558 is used.
1559 Multiple
1560 .Ar utility
1561 arguments are treated as separate utilities.
1562 .Pp
1563 See also
1564 .Sx \&Rv .
1565 .Ss \&Fa
1566 Function argument.
1567 Its syntax is as follows:
1568 .Bd -ragged -offset indent
1569 .Pf \. Sx \&Fa
1570 .Op Cm argtype
1571 .Cm argname
1572 .Ed
1573 .Pp
1574 This may be invoked for names with or without the corresponding type.

```

```

1575 It is also used to specify the field name of a structure.
1576 Most often, the
1577 .Sx \&Fa
1578 macro is used in the
1579 .Em SYNOPSIS
1580 within
1581 .Sx \&Fo
1582 section when documenting multi-line function prototypes.
1583 If invoked with multiple arguments, the arguments are separated by a
1584 comma.
1585 Furthermore, if the following macro is another
1586 .Sx \&Fa ,
1587 the last argument will also have a trailing comma.
1588 .Pp
1589 Examples:
1590 .Dl \&Fa \((dqconst char *p\((dq
1591 .Dl \&Fa \((dqint a\((dq \((dqint b\((dq \((dqint c\((dq
1592 .Dl \&Fa foo
1593 .Pp
1594 See also
1595 .Sx \&Fo .
1596 .Ss \&Fc
1597 End a function context started by
1598 .Sx \&Fo .
1599 .Ss \&Fd
1600 Historically used to document include files.
1601 This usage has been deprecated in favour of
1602 .Sx \&In .
1603 Do not use this macro.
1604 .Pp
1605 See also
1606 .Sx MANUAL STRUCTURE
1607 and
1608 .Sx \&In .
1609 .Ss \&Fl
1610 Command-line flag or option.
1611 Used when listing arguments to command-line utilities.
1612 Prints a fixed-width hyphen
1613 .Sq \-
1614 directly followed by each argument.
1615 If no arguments are provided, a hyphen is printed followed by a space.
1616 If the argument is a macro, a hyphen is prefixed to the subsequent macro
1617 output.
1618 .Pp
1619 Examples:
1620 .Dl ".Fl R Op Fl H | L | P"
1621 .Dl ".Op Fl lAaCcdFfgHhikLlmnopqRrSsTtux"
1622 .Dl ".Fl type Cm d Fl name Pa CVS"
1623 .Dl ".Fl Ar signal_number"
1624 .Dl ".Fl o Fl"
1625 .Pp
1626 See also
1627 .Sx \&Cm .
1628 .Ss \&Fn
1629 A function name.
1630 Its syntax is as follows:
1631 .Bd -ragged -offset indent
1632 .Pf \. Ns Sx \&Fn
1633 .Op Ar functype
1634 .Ar funcname
1635 .Op Oo Ar argtype Oc Ar argname
1636 .Ed
1637 .Pp
1638 Function arguments are surrounded in parenthesis and
1639 are delimited by commas.
1640 If no arguments are specified, blank parenthesis are output.

```

```

1641 In the
1642 .Em SYNOPSIS
1643 section, this macro starts a new output line,
1644 and a blank line is automatically inserted between function definitions.
1645 .Pp
1646 Examples:
1647 .Dl \&Fn \((dqint funcname\((dq \((dqint arg0\((dq \((dqint arg1\((dq
1648 .Dl \&Fn funcname \((dqint arg0\((dq
1649 .Dl \&Fn funcname arg0
1650 .Pp
1651 .Bd -literal -offset indent -compact
1652 \&Ft functype
1653 \&Fn funcname
1654 .Ed
1655 .Pp
1656 When referring to a function documented in another manual page, use
1657 .Sx \&Xr
1658 instead.
1659 See also
1660 .Sx MANUAL STRUCTURE ,
1661 .Sx \&Fo ,
1662 and
1663 .Sx \&Ft .
1664 .Ss \&Fo
1665 Begin a function block.
1666 This is a multi-line version of
1667 .Sx \&Fn .
1668 Its syntax is as follows:
1669 .Pp
1670 .Dl Pf \. Sx \&Fo Ar funcname
1671 .Pp
1672 Invocations usually occur in the following context:
1673 .Bd -ragged -offset indent
1674 .Pf \. Sx \&Ft Ar functype
1675 .br
1676 .Pf \. Sx \&Fo Ar funcname
1677 .br
1678 .Pf \. Sx \&Fa Oo Ar argtype Oc Ar argname
1679 .br
1680 \&.\.
1681 .br
1682 .Pf \. Sx \&Fc
1683 .Ed
1684 .Pp
1685 A
1686 .Sx \&Fo
1687 scope is closed by
1688 .Sx \&Fc .
1689 .Pp
1690 See also
1691 .Sx MANUAL STRUCTURE ,
1692 .Sx \&Fa ,
1693 .Sx \&Fc ,
1694 and
1695 .Sx \&Ft .
1696 .Ss \&Fr
1697 This macro is obsolete and not implemented in
1698 .Xr mandoc 1 .
1699 .Pp
1700 It was used to show function return values.
1701 The syntax was:
1702 .Pp
1703 .Dl Pf . Sx \&Fr Ar value
1704 .Ss \&Ft
1705 A function type.
1706 Its syntax is as follows:

```

```

1707 .Pp
1708 .Dl Pf \. Sx \&Ft Ar functype
1709 .Pp
1710 In the
1711 .Em SYNOPSIS
1712 section, a new output line is started after this macro.
1713 .Pp
1714 Examples:
1715 .Dl \&Ft int
1716 .Bd -literal -offset indent -compact
1717 \&Ft functype
1718 \&Fn funcname
1719 .Ed
1720 .Pp
1721 See also
1722 .Sx MANUAL STRUCTURE ,
1723 .Sx \&Fn ,
1724 and
1725 .Sx \&Fo .
1726 .Ss \&Fx
1727 Format the
1728 .Fx
1729 version provided as an argument, or a default value
1730 if no argument is provided.
1731 .Pp
1732 Examples:
1733 .Dl \&Fx 7.1
1734 .Dl \&Fx
1735 .Pp
1736 See also
1737 .Sx \&At ,
1738 .Sx \&Bsx ,
1739 .Sx \&Bx ,
1740 .Sx \&Dx ,
1741 .Sx \&Nx ,
1742 .Sx \&Ox ,
1743 and
1744 .Sx \&Ux .
1745 .Ss \&Hf
1746 This macro is not implemented in
1747 .Xr mandoc 1 .
1748 .Pp
1749 It was used to include the contents of a (header) file literally.
1750 The syntax was:
1751 .Pp
1752 .Dl Pf . Sx \&Hf Ar filename
1753 .Ss \&Ic
1754 Designate an internal or interactive command.
1755 This is similar to
1756 .Sx \&Cm
1757 but used for instructions rather than values.
1758 .Pp
1759 Examples:
1760 .Dl \&Ic :wq
1761 .Dl \&Ic hash
1762 .Dl \&Ic alias
1763 .Pp
1764 Note that using
1765 .Sx \&Bd Fl literal
1766 or
1767 .Sx \&Dl
1768 is preferred for displaying code; the
1769 .Sx \&Ic
1770 macro is used when referring to specific instructions.
1771 .Ss \&In
1772 An

```

```

1773 .Dq include
1774 file.
1775 When invoked as the first macro on an input line in the
1776 .Em SYNOPSIS
1777 section, the argument is displayed in angle brackets
1778 and preceded by
1779 .Dq #include ,
1780 and a blank line is inserted in front if there is a preceding
1781 function declaration.
1782 This is most often used in section 2, 3, and 9 manual pages.
1783 .Pp
1784 Examples:
1785 .Dl \&In sys/types.h
1786 .Pp
1787 See also
1788 .Sx MANUAL STRUCTURE .
1789 .Ss \&It
1790 A list item.
1791 The syntax of this macro depends on the list type.
1792 .Pp
1793 Lists
1794 of type
1795 .Fl hang ,
1796 .Fl ohang ,
1797 .Fl inset ,
1798 and
1799 .Fl diag
1800 have the following syntax:
1801 .Pp
1802 .Dl Pf \. Sx \&It Ar args
1803 .Pp
1804 Lists of type
1805 .Fl bullet ,
1806 .Fl dash ,
1807 .Fl enum ,
1808 .Fl hyphen
1809 and
1810 .Fl item
1811 have the following syntax:
1812 .Pp
1813 .Dl Pf \. Sx \&It
1814 .Pp
1815 with subsequent lines interpreted within the scope of the
1816 .Sx \&It
1817 until either a closing
1818 .Sx \&El
1819 or another
1820 .Sx \&It .
1821 .Pp
1822 The
1823 .Fl tag
1824 list has the following syntax:
1825 .Pp
1826 .Dl Pf \. Sx \&It Op Cm args
1827 .Pp
1828 Subsequent lines are interpreted as with
1829 .Fl bullet
1830 and family.
1831 The line arguments correspond to the list's left-hand side; body
1832 arguments correspond to the list's contents.
1833 .Pp
1834 The
1835 .Fl column
1836 list is the most complicated.
1837 Its syntax is as follows:
1838 .Pp

```

```

1839 .Dl Pf \. Sx \&It Ar cell Op <TAB> Ar cell ...
1840 .Dl Pf \. Sx \&It Ar cell Op Sx \&Ta Ar cell ...
1841 .Pp
1842 The arguments consist of one or more lines of text and macros
1843 representing a complete table line.
1844 Cells within the line are delimited by tabs or by the special
1845 .Sx \&Ta
1846 block macro.
1847 The tab cell delimiter may only be used within the
1848 .Sx \&It
1849 line itself; on following lines, only the
1850 .Sx \&Ta
1851 macro can be used to delimit cells, and
1852 .Sx \&Ta
1853 is only recognised as a macro when called by other macros,
1854 not as the first macro on a line.
1855 .Pp
1856 Note that quoted strings may span tab-delimited cells on an
1857 .Sx \&It
1858 line.
1859 For example,
1860 .Pp
1861 .Dl .It \dqcoll ; <TAB> col2 ; \dq \& ;
1862 .Pp
1863 will preserve the semicolon whitespace except for the last.
1864 .Pp
1865 See also
1866 .Sx \&Bl .
1867 .Ss \&Lb
1868 Specify a library.
1869 The syntax is as follows:
1870 .Pp
1871 .Dl Pf \. Sx \&Lb Ar library
1872 .Pp
1873 The
1874 .Ar library
1875 parameter may be a system library, such as
1876 .Cm libz
1877 or
1878 .Cm libpam ,
1879 in which case a small library description is printed next to the linker
1880 invocation; or a custom library, in which case the library name is
1881 printed in quotes.
1882 This is most commonly used in the
1883 .Em SYNOPSIS
1884 section as described in
1885 .Sx MANUAL STRUCTURE .
1886 .Pp
1887 Examples:
1888 .Dl \&Lb libz
1889 .Dl \&Lb mdoc
1890 .Ss \&Li
1891 Denotes text that should be in a
1892 .Li literal
1893 font mode.
1894 Note that this is a presentation term and should not be used for
1895 stylistically decorating technical terms.
1896 .Pp
1897 On terminal output devices, this is often indistinguishable from
1898 normal text.
1899 .Pp
1900 See also
1901 .Sx \&Bf ,
1902 .Sx \&Em ,
1903 .Sx \&No ,
1904 and

```

```

1905 .Sx \&Sy .
1906 .Ss \&Lk
1907 Format a hyperlink.
1908 Its syntax is as follows:
1909 .Pp
1910 .Dl Pf \. Sx \&Lk Ar uri Op Ar name
1911 .Pp
1912 Examples:
1913 .Dl \&Lk http://bsd.lv \dqThe BSD.lv Project\dq
1914 .Dl \&Lk http://bsd.lv
1915 .Pp
1916 See also
1917 .Sx \&Mt .
1918 .Ss \&Lp
1919 Synonym for
1920 .Sx \&Pp .
1921 .Ss \&Ms
1922 Display a mathematical symbol.
1923 Its syntax is as follows:
1924 .Pp
1925 .Dl Pf \. Sx \&Ms Ar symbol
1926 .Pp
1927 Examples:
1928 .Dl \&Ms sigma
1929 .Dl \&Ms aleph
1930 .Ss \&Mt
1931 Format a
1932 .Dq mailto:
1933 hyperlink.
1934 Its syntax is as follows:
1935 .Pp
1936 .Dl Pf \. Sx \&Mt Ar address
1937 .Pp
1938 Examples:
1939 .Dl \&Mt discuss@manpages.bsd.lv
1940 .Ss \&Nd
1941 A one line description of the manual's content.
1942 This may only be invoked in the
1943 .Em SYNOPSIS
1944 section subsequent the
1945 .Sx \&Nm
1946 macro.
1947 .Pp
1948 Examples:
1949 .Dl Pf . Sx \&Nd mdoc language reference
1950 .Dl Pf . Sx \&Nd format and display UNIX manuals
1951 .Pp
1952 The
1953 .Sx \&Nd
1954 macro technically accepts child macros and terminates with a subsequent
1955 .Sx \&Sh
1956 invocation.
1957 Do not assume this behaviour: some
1958 .Xr whatis 1
1959 database generators are not smart enough to parse more than the line
1960 arguments and will display macros verbatim.
1961 .Pp
1962 See also
1963 .Sx \&Nm .
1964 .Ss \&Nm
1965 The name of the manual page, or \em in particular in section 1
1966 and 1M pages \em of an additional command or feature documented in
1967 the manual page.
1968 When first invoked, the
1969 .Sx \&Nm
1970 macro expects a single argument, the name of the manual page.

```

1971 Usually, the first invocation happens in the
 1972 **.Em** NAME
 1973 section of the page.
 1974 The specified name will be remembered and used whenever the macro is
 1975 called again without arguments later in the page.
 1976 The
 1977 **.Sx** \&Nm
 1978 macro uses
 1979 **.Sx** Block full-implicit
 1980 semantics when invoked as the first macro on an input line in the
 1981 **.Em** SYNOPSIS
 1982 section; otherwise, it uses ordinary
 1983 **.Sx** In-line
 1984 semantics.
 1985 **.Pp**
 1986 Examples:
 1987 **.Bd** -literal -offset indent
 1988 \&Sh SYNOPSIS
 1989 \&.Nm cat
 1990 \&.Op Fl benstuv
 1991 \&.Op Ar
 1992 **.Ed**
 1993 **.Pp**
 1994 In the
 1995 **.Em** SYNOPSIS
 1996 of section 2, 3 and 9 manual pages, use the
 1997 **.Sx** \&Fn
 1998 macro rather than
 1999 **.Sx** \&Nm
 2000 to mark up the name of the manual page.
 2001 **.Ss** \&No
 2002 Normal text.
 2003 Closes the scope of any preceding in-line macro.
 2004 When used after physical formatting macros like
 2005 **.Sx** \&Em
 2006 or
 2007 **.Sx** \&Sy ,
 2008 switches back to the standard font face and weight.
 2009 Can also be used to embed plain text strings in macro lines
 2010 using semantic annotation macros.
 2011 **.Pp**
 2012 Examples:
 2013 **.Dl** ".Em italic , Sy bold , No and roman"
 2014 **.Pp**
 2015 **.Bd** -literal -offset indent -compact
 2016 \&.Sm off
 2017 \&.Cm :C No / Ar pattern No / Ar replacement No /
 2018 \&.Sm on
 2019 **.Ed**
 2020 **.Pp**
 2021 See also
 2022 **.Sx** \&Em ,
 2023 **.Sx** \&Li ,
 2024 and
 2025 **.Sx** \&Sy .
 2026 **.Ss** \&Ns
 2027 Suppress a space between the output of the preceding macro
 2028 and the following text or macro.
 2029 Following invocation, input is interpreted as normal text
 2030 just like after an
 2031 **.Sx** \&No
 2032 macro.
 2033 **.Pp**
 2034 This has no effect when invoked at the start of a macro line.
 2035 **.Pp**
 2036 Examples:

2037 **.Dl** ".Ar name Ns = Ns Ar value"
 2038 **.Dl** ".Cm :M Ns Ar pattern"
 2039 **.Dl** ".Fl o Ns Ar output"
 2040 **.Pp**
 2041 See also
 2042 **.Sx** \&No
 2043 and
 2044 **.Sx** \&Sm .
 2045 **.Ss** \&Nx
 2046 Format the
 2047 **.Nx**
 2048 version provided as an argument, or a default value if
 2049 no argument is provided.
 2050 **.Pp**
 2051 Examples:
 2052 **.Dl** \&.Nx 5.01
 2053 **.Dl** \&.Nx
 2054 **.Pp**
 2055 See also
 2056 **.Sx** \&At ,
 2057 **.Sx** \&Bsx ,
 2058 **.Sx** \&Bx ,
 2059 **.Sx** \&Dx ,
 2060 **.Sx** \&Fx ,
 2061 **.Sx** \&Ox ,
 2062 and
 2063 **.Sx** \&Ux .
 2064 **.Ss** \&Oc
 2065 Close multi-line
 2066 **.Sx** \&Oo
 2067 context.
 2068 **.Ss** \&Oo
 2069 Multi-line version of
 2070 **.Sx** \&Op .
 2071 **.Pp**
 2072 Examples:
 2073 **.Bd** -literal -offset indent -compact
 2074 \&.Oo
 2075 \&.Op Fl flag Ns Ar value
 2076 \&.Oc
 2077 **.Ed**
 2078 **.Ss** \&Op
 2079 Optional part of a command line.
 2080 Prints the argument(s) in brackets.
 2081 This is most often used in the
 2082 **.Em** SYNOPSIS
 2083 section of section 1 and 1M manual pages.
 2084 **.Pp**
 2085 Examples:
 2086 **.Dl** \&.Op \&Fl a \&Ar b
 2087 **.Dl** \&.Op \&Ar a | b
 2088 **.Pp**
 2089 See also
 2090 **.Sx** \&Oo .
 2091 **.Ss** \&Os
 2092 Document operating system version.
 2093 This is the mandatory third macro of
 2094 any
 2095 **.Nm**
 2096 file.
 2097 Its syntax is as follows:
 2098 **.Pp**
 2099 **.Dl** Pf \. Sx \&Os Op Ar system Op Ar version
 2100 **.Pp**
 2101 The optional
 2102 **.Ar** system

2103 parameter specifies the relevant operating system or environment.
 2104 Left unspecified, it defaults to the local operating system version.
 2105 This is the suggested form.
 2106 .Pp
 2107 Examples:
 2108 .Dl \&.Os
 2109 .Dl \&.Os KTH/CSC/TCS
 2110 .Dl \&.Os BSD 4.3
 2111 .Pp
 2112 See also
 2113 .Sx \&Dd
 2114 and
 2115 .Sx \&Dt .
 2116 .Ss \&Ot
 2117 This macro is obsolete and not implemented in
 2118 .Xr mandoc 1 .
 2119 .Pp
 2120 Historical
 2121 .Xr mdoc 5
 2122 packages described it as
 2123 .Dq "old function type (FORTRAN)" .
 2124 .Ss \&Ox
 2125 Format the
 2126 .Ox
 2127 version provided as an argument, or a default value
 2128 if no argument is provided.
 2129 .Pp
 2130 Examples:
 2131 .Dl \&.Ox 4.5
 2132 .Dl \&.Ox
 2133 .Pp
 2134 See also
 2135 .Sx \&At ,
 2136 .Sx \&BSx ,
 2137 .Sx \&Bx ,
 2138 .Sx \&Dx ,
 2139 .Sx \&Fx ,
 2140 .Sx \&Nx ,
 2141 and
 2142 .Sx \&Ux .
 2143 .Ss \&Pa
 2144 An absolute or relative file system path, or a file or directory name.
 2145 If an argument is not provided, the character
 2146 .Sq \(\ti
 2147 is used as a default.
 2148 .Pp
 2149 Examples:
 2150 .Dl \&.Pa /usr/bin/mandoc
 2151 .Dl \&.Pa /usr/share/man/man5/mdoc.5
 2152 .Pp
 2153 See also
 2154 .Sx \&Lk .
 2155 .Ss \&Pc
 2156 Close parenthesised context opened by
 2157 .Sx \&Po .
 2158 .Ss \&Pf
 2159 Removes the space between its argument
 2160 .Pq Dq prefix
 2161 and the following macro.
 2162 Its syntax is as follows:
 2163 .Pp
 2164 .Dl .Pf Ar prefix macro arguments ...
 2165 .Pp
 2166 This is equivalent to:
 2167 .Pp
 2168 .Dl .No Ar prefix No \&Ns Ar macro arguments ...

2169 .Pp
 2170 Examples:
 2171 .Dl ".Pf \$ Ar variable_name"
 2172 .Dl ".Pf 0x Ar hex_digits"
 2173 .Pp
 2174 See also
 2175 .Sx \&Ns
 2176 and
 2177 .Sx \&Sm .
 2178 .Ss \&Po
 2179 Multi-line version of
 2180 .Sx \&Pq .
 2181 .Ss \&Pp
 2182 Break a paragraph.
 2183 This will assert vertical space between prior and subsequent macros
 2184 and/or text.
 2185 .Pp
 2186 Paragraph breaks are not needed before or after
 2187 .Sx \&Sh
 2188 or
 2189 .Sx \&Ss
 2190 macros or before displays
 2191 .Pq Sx \&Bd
 2192 or lists
 2193 .Pq Sx \&Bl
 2194 unless the
 2195 .Fl compact
 2196 flag is given.
 2197 .Ss \&Pq
 2198 Parenthesised enclosure.
 2199 .Pp
 2200 See also
 2201 .Sx \&Po .
 2202 .Ss \&Qc
 2203 Close quoted context opened by
 2204 .Sx \&Qo .
 2205 .Ss \&Ql
 2206 Format a single-quoted literal.
 2207 See also
 2208 .Sx \&Qq
 2209 and
 2210 .Sx \&Sq .
 2211 .Ss \&Qo
 2212 Multi-line version of
 2213 .Sx \&Qq .
 2214 .Ss \&Qq
 2215 Encloses its arguments in
 2216 .Qq typewriter
 2217 double-quotes.
 2218 Consider using
 2219 .Sx \&Dq .
 2220 .Pp
 2221 See also
 2222 .Sx \&Dq ,
 2223 .Sx \&Sq ,
 2224 and
 2225 .Sx \&Qo .
 2226 .Ss \&Re
 2227 Close an
 2228 .Sx \&Rs
 2229 block.
 2230 Does not have any tail arguments.
 2231 .Ss \&Rs
 2232 Begin a bibliographic
 2233 .Pq Dq reference
 2234 block.

```

2235 Does not have any head arguments.
2236 The block macro may only contain
2237 .Sx \&A ,
2238 .Sx \&B ,
2239 .Sx \&C ,
2240 .Sx \&D ,
2241 .Sx \&I ,
2242 .Sx \&J ,
2243 .Sx \&N ,
2244 .Sx \&O ,
2245 .Sx \&P ,
2246 .Sx \&Q ,
2247 .Sx \&R ,
2248 .Sx \&T ,
2249 .Sx \&U ,
2250 and
2251 .Sx \&V
2252 child macros (at least one must be specified).
2253 .Pp
2254 Examples:
2255 .Bd -literal -offset indent -compact
2256 \&.Rs
2257 \&.%A J. E. Hopcroft
2258 \&.%A J. D. Ullman
2259 \&.%B Introduction to Automata Theory, Languages, and Computation
2260 \&.%I Addison-Wesley
2261 \&.%C Reading, Massachusettes
2262 \&.%D 1979
2263 \&.Re
2264 .Ed
2265 .Pp
2266 If an
2267 .Sx \&Rs
2268 block is used within a SEE ALSO section, a vertical space is asserted
2269 before the rendered output, else the block continues on the current
2270 line.
2271 .Ss \&Rv
2272 Insert a standard sentence regarding a function call's return value of 0
2273 on success and \-1 on error, with the
2274 .Va errno
2275 libc global variable set on error.
2276 Its syntax is as follows:
2277 .Pp
2278 .Dl Pf \. Sx \&Rv Fl std Op Ar function ...
2279 .Pp
2280 If
2281 .Ar function
2282 is not specified, the document's name set by
2283 .Sx \&Nm
2284 is used.
2285 Multiple
2286 .Ar function
2287 arguments are treated as separate functions.
2288 .Pp
2289 See also
2290 .Sx \&Ex .
2291 .Ss \&Sc
2292 Close single-quoted context opened by
2293 .Sx \&So .
2294 .Ss \&Sh
2295 Begin a new section.
2296 For a list of conventional manual sections, see
2297 .Sx MANUAL STRUCTURE .
2298 These sections should be used unless it's absolutely necessary that
2299 custom sections be used.
2300 .Pp

```

```

2301 Section names should be unique so that they may be keyed by
2302 .Sx \&Sx .
2303 Although this macro is parsed, it should not consist of child node or it
2304 may not be linked with
2305 .Sx \&Sx .
2306 .Pp
2307 See also
2308 .Sx \&Pp ,
2309 .Sx \&Ss ,
2310 and
2311 .Sx \&Sx .
2312 .Ss \&Sm
2313 Switches the spacing mode for output generated from macros.
2314 Its syntax is as follows:
2315 .Pp
2316 .Dl Pf \. Sx \&Sm Cm on | off
2317 .Pp
2318 By default, spacing is
2319 .Cm on .
2320 When switched
2321 .Cm off ,
2322 no white space is inserted between macro arguments and between the
2323 output generated from adjacent macros, but text lines
2324 still get normal spacing between words and sentences.
2325 .Ss \&So
2326 Multi-line version of
2327 .Sx \&Sq .
2328 .Ss \&Sq
2329 Encloses its arguments in
2330 .Sq typewriter
2331 single-quotes.
2332 .Pp
2333 See also
2334 .Sx \&Dq ,
2335 .Sx \&Qq ,
2336 and
2337 .Sx \&So .
2338 .Ss \&Ss
2339 Begin a new subsection.
2340 Unlike with
2341 .Sx \&Sh ,
2342 there is no convention for the naming of subsections.
2343 Except
2344 .Em DESCRIPTION ,
2345 the conventional sections described in
2346 .Sx MANUAL STRUCTURE
2347 rarely have subsections.
2348 .Pp
2349 Sub-section names should be unique so that they may be keyed by
2350 .Sx \&Sx .
2351 Although this macro is parsed, it should not consist of child node or it
2352 may not be linked with
2353 .Sx \&Sx .
2354 .Pp
2355 See also
2356 .Sx \&Pp ,
2357 .Sx \&Sh ,
2358 and
2359 .Sx \&Sx .
2360 .Ss \&St
2361 Replace an abbreviation for a standard with the full form.
2362 The following standards are recognised:
2363 .Pp
2364 .Bl -tag -width "-p1003.1g-2000X" -compact
2365 .It \-p1003.1-88
2366 .St -p1003.1-88

```

```

2367 .It \-p1003.1-90
2368 .St -p1003.1-90
2369 .It \-p1003.1-96
2370 .St -p1003.1-96
2371 .It \-p1003.1-2001
2372 .St -p1003.1-2001
2373 .It \-p1003.1-2004
2374 .St -p1003.1-2004
2375 .It \-p1003.1-2008
2376 .St -p1003.1-2008
2377 .It \-p1003.1
2378 .St -p1003.1
2379 .It \-p1003.1b
2380 .St -p1003.1b
2381 .It \-p1003.1b-93
2382 .St -p1003.1b-93
2383 .It \-p1003.1c-95
2384 .St -p1003.1c-95
2385 .It \-p1003.1g-2000
2386 .St -p1003.1g-2000
2387 .It \-p1003.1i-95
2388 .St -p1003.1i-95
2389 .It \-p1003.2-92
2390 .St -p1003.2-92
2391 .It \-p1003.2a-92
2392 .St -p1003.2a-92
2393 .It \-p1387.2-95
2394 .St -p1387.2-95
2395 .It \-p1003.2
2396 .St -p1003.2
2397 .It \-p1387.2
2398 .St -p1387.2
2399 .It \-isoC
2400 .St -isoC
2401 .It \-isoC-90
2402 .St -isoC-90
2403 .It \-isoC-amd1
2404 .St -isoC-amd1
2405 .It \-isoC-tcor1
2406 .St -isoC-tcor1
2407 .It \-isoC-tcor2
2408 .St -isoC-tcor2
2409 .It \-isoC-99
2410 .St -isoC-99
2411 .It \-isoC-2011
2412 .St -isoC-2011
2413 .It \-iso9945-1-90
2414 .St -iso9945-1-90
2415 .It \-iso9945-1-96
2416 .St -iso9945-1-96
2417 .It \-iso9945-2-93
2418 .St -iso9945-2-93
2419 .It \-ansiC
2420 .St -ansiC
2421 .It \-ansiC-89
2422 .St -ansiC-89
2423 .It \-ansiC-99
2424 .St -ansiC-99
2425 .It \-ieee754
2426 .St -ieee754
2427 .It \-iso8802-3
2428 .St -iso8802-3
2429 .It \-iso8601
2430 .St -iso8601
2431 .It \-ieee1275-94
2432 .St -ieee1275-94

```

```

2433 .It \-xpg3
2434 .St -xpg3
2435 .It \-xpg4
2436 .St -xpg4
2437 .It \-xpg4.2
2438 .St -xpg4.2
2439 .It \-xpg4.3
2440 .St -xpg4.3
2441 .It \-xbd5
2442 .St -xbd5
2443 .It \-xcu5
2444 .St -xcu5
2445 .It \-xsh5
2446 .St -xsh5
2447 .It \-xns5
2448 .St -xns5
2449 .It \-xns5.2
2450 .St -xns5.2
2451 .It \-xns5.2d2.0
2452 .St -xns5.2d2.0
2453 .It \-xcurses4.2
2454 .St -xcurses4.2
2455 .It \-susv2
2456 .St -susv2
2457 .It \-susv3
2458 .St -susv3
2459 .It \-svid4
2460 .St -svid4
2461 .El
2462 .Ss \&Sx
2463 Reference a section or subsection in the same manual page.
2464 The referenced section or subsection name must be identical to the
2465 enclosed argument, including whitespace.
2466 .Pp
2467 Examples:
2468 .Dl \&Sx MANUAL STRUCTURE
2469 .Pp
2470 See also
2471 .Sx \&Sh
2472 and
2473 .Sx \&Ss .
2474 .Ss \&Sy
2475 Format enclosed arguments in symbolic
2476 .Pq Dq boldface .
2477 Note that this is a presentation term and should not be used for
2478 stylistically decorating technical terms.
2479 .Pp
2480 See also
2481 .Sx \&Bf ,
2482 .Sx \&Em ,
2483 .Sx \&Li ,
2484 and
2485 .Sx \&No .
2486 .Ss \&Ta
2487 Table cell separator in
2488 .Sx \&B1 F1 column
2489 lists; can only be used below
2490 .Sx \&It .
2491 .Ss \&Tn
2492 Format a tradename.
2493 .Pp
2494 Since this macro is often implemented to use a small caps font,
2495 it has historically been used for acronyms (like ASCII) as well.
2496 Such usage is not recommended because it would use the same macro
2497 sometimes for semantical annotation, sometimes for physical formatting.
2498 .Pp

```



```

2499 Examples:
2500 .Dl \&.Tn IBM
2501 .Ss \&Ud
2502 Prints out
2503 .Dq currently under development.
2504 .Ss \&Ux
2505 Format the UNIX name.
2506 Accepts no argument.
2507 .Pp
2508 Examples:
2509 .Dl \&.Ux
2510 .Pp
2511 See also
2512 .Sx \&At ,
2513 .Sx \&Bsx ,
2514 .Sx \&Bx ,
2515 .Sx \&Dx ,
2516 .Sx \&Fx ,
2517 .Sx \&Nx ,
2518 and
2519 .Sx \&Ox .
2520 .Ss \&Va
2521 A variable name.
2522 .Pp
2523 Examples:
2524 .Dl \&.Va foo
2525 .Dl \&.Va const char *bar ;
2526 .Ss \&Vt
2527 A variable type.
2528 This is also used for indicating global variables in the
2529 .Em SYNOPSIS
2530 section, in which case a variable name is also specified.
2531 Note that it accepts
2532 .Sx Block partial-implicit
2533 syntax when invoked as the first macro on an input line in the
2534 .Em SYNOPSIS
2535 section, else it accepts ordinary
2536 .Sx In-line
2537 syntax.
2538 In the former case, this macro starts a new output line,
2539 and a blank line is inserted in front if there is a preceding
2540 function definition or include directive.
2541 .Pp
2542 Note that this should not be confused with
2543 .Sx \&Ft ,
2544 which is used for function return types.
2545 .Pp
2546 Examples:
2547 .Dl \&.Vt unsigned char
2548 .Dl \&.Vt extern const char * const sys_signame[] \&;
2549 .Pp
2550 See also
2551 .Sx MANUAL STRUCTURE
2552 and
2553 .Sx \&Va .
2554 .Ss \&Xc
2555 Close a scope opened by
2556 .Sx \&Xo .
2557 .Ss \&Xo .
2558 Extend the header of an
2559 .Sx \&It
2560 macro or the body of a partial-implicit block macro
2561 beyond the end of the input line.
2562 This macro originally existed to work around the 9-argument limit
2563 of historic
2564 .Xr roff 5 .

```

```

2565 .Ss \&Xr
2566 Link to another manual
2567 .Pq Qq cross-reference .
2568 Its syntax is as follows:
2569 .Pp
2570 .Dl Pf \. Sx \&Xr Ar name section
2571 .Pp
2572 The
2573 .Ar name
2574 and
2575 .Ar section
2576 are the name and section of the linked manual.
2577 If
2578 .Ar section
2579 is followed by non-punctuation, an
2580 .Sx \&Ns
2581 is inserted into the token stream.
2582 This behaviour is for compatibility with
2583 GNU troff.
2584 .Pp
2585 Examples:
2586 .Dl \&.Xr mandoc 1
2587 .Dl \&.Xr mandoc 1 \&;
2588 .Dl \&.Xr mandoc 1 \&Ns s behaviour
2589 .Ss \&br
2590 Emits a line-break.
2591 This macro should not be used; it is implemented for compatibility with
2592 historical manuals.
2593 .Pp
2594 Consider using
2595 .Sx \&Pp
2596 in the event of natural paragraph breaks.
2597 .Ss \&sp
2598 Emits vertical space.
2599 This macro should not be used; it is implemented for compatibility with
2600 historical manuals.
2601 Its syntax is as follows:
2602 .Pp
2603 .Dl Pf \. Sx \&sp Op Ar height
2604 .Pp
2605 The
2606 .Ar height
2607 argument must be formatted as described in
2608 .Sx Scaling Widths .
2609 If unspecified,
2610 .Sx \&sp
2611 asserts a single vertical space.
2612 .Sh MACRO SYNTAX
2613 The syntax of a macro depends on its classification.
2614 In this section,
2615 .Sq \-arg
2616 refers to macro arguments, which may be followed by zero or more
2617 .Sq parm
2618 parameters;
2619 .Sq \&Yo
2620 opens the scope of a macro; and if specified,
2621 .Sq \&Yc
2622 closes it out.
2623 .Pp
2624 The
2625 .Em Callable
2626 column indicates that the macro may also be called by passing its name
2627 as an argument to another macro.
2628 For example,
2629 .Sq \&.Op \&F1 O \&Ar file
2630 produces

```

```

2631 .Sq Op Fl O Ar file .
2632 To prevent a macro call and render the macro name literally,
2633 escape it by prepending a zero-width space,
2634 .Sq \e& .
2635 For example,
2636 .Sq \&Op \e&Fl O
2637 produces
2638 .Sq Op \&Fl O .
2639 If a macro is not callable but its name appears as an argument
2640 to another macro, it is interpreted as opaque text.
2641 For example,
2642 .Sq \&.Fl \&Sh
2643 produces
2644 .Sq Fl \&Sh .
2645 .Pp
2646 The
2647 .Em Parsed
2648 column indicates whether the macro may call other macros by receiving
2649 their names as arguments.
2650 If a macro is not parsed but the name of another macro appears
2651 as an argument, it is interpreted as opaque text.
2652 .Pp
2653 The
2654 .Em Scope
2655 column, if applicable, describes closure rules.
2656 .Ss Block full-explicit
2657 Multi-line scope closed by an explicit closing macro.
2658 All macros contains bodies; only
2659 .Sx \&Bf
2660 and
2661 .Pq optionally
2662 .Sx \&Bl
2663 contain a head.
2664 .Bd -literal -offset indent
2665 \&.Yo \(\lB\ -arg \(\lBparm...\(rB\ (rB \(\lBhead...\(rB
2666 \(\lBbody...\(rB
2667 \&.Yc
2668 .Ed
2669 .Bl -column "MacroX" "CallableX" "ParsedX" "closed by XXX" -offset indent
2670 .It Em Macro Ta Em Callable Ta Em Parsed Ta Em Scope
2671 .It Sx \&Bd Ta \&No Ta \&No Ta closed by Sx \&Ed
2672 .It Sx \&Bf Ta \&No Ta \&No Ta closed by Sx \&Ef
2673 .It Sx \&Bk Ta \&No Ta \&No Ta closed by Sx \&Ek
2674 .It Sx \&Bl Ta \&No Ta \&No Ta closed by Sx \&El
2675 .It Sx \&Ed Ta \&No Ta \&No Ta opened by Sx \&Bd
2676 .It Sx \&Ef Ta \&No Ta \&No Ta opened by Sx \&Bf
2677 .It Sx \&Ek Ta \&No Ta \&No Ta opened by Sx \&Bk
2678 .It Sx \&El Ta \&No Ta \&No Ta opened by Sx \&Bl
2679 .El
2680 .Ss Block full-implicit
2681 Multi-line scope closed by end-of-file or implicitly by another macro.
2682 All macros has bodies; some
2683 .Po
2684 .Sx \&It Fl bullet ,
2685 .Fl hyphen ,
2686 .Fl dash ,
2687 .Fl enum ,
2688 .Fl item
2689 .Pc
2690 don't have heads; only one
2691 .Po
2692 .Sx \&It
2693 in
2694 .Sx \&Bl Fl column
2695 .Pc
2696 has multiple heads.

```

```

2697 .Bd -literal -offset indent
2698 \&.Yo \(\lB\ -arg \(\lBparm...\(rB\ (rB \(\lBhead...\(lB\Ta head...\(rB\ (rB
2699 \(\lBbody...\(rB
2700 .Ed
2701 .Bl -column "MacroX" "CallableX" "ParsedX" "closed by XXXXXXXXXXXX" -offset inden
2702 .It Em Macro Ta Em Callable Ta Em Parsed Ta Em Scope
2703 .It Sx \&It Ta \&No Ta Yes Ta closed by Sx \&It , Sx \&El
2704 .It Sx \&Nd Ta \&No Ta \&No Ta closed by Sx \&Sh
2705 .It Sx \&Nm Ta \&No Ta Yes Ta closed by Sx \&Nm , Sx \&Sh , Sx \&Ss
2706 .It Sx \&Sh Ta \&No Ta Yes Ta closed by Sx \&Sh
2707 .It Sx \&Ss Ta \&No Ta Yes Ta closed by Sx \&Sh , Sx \&Ss
2708 .El
2709 .Pp
2710 Note that the
2711 .Sx \&Nm
2712 macro is a
2713 .Sx Block full-implicit
2714 macro only when invoked as the first macro
2715 in a
2716 .Em SYNOPSIS
2717 section line, else it is
2718 .Sx In-line .
2719 .Ss Block partial-explicit
2720 Like block full-explicit, but also with single-line scope.
2721 Each has at least a body and, in limited circumstances, a head
2722 .Po
2723 .Sx \&Fo ,
2724 .Sx \&Eo
2725 .Pc
2726 and/or tail
2727 .Pq Sx \&Ec .
2728 .Bd -literal -offset indent
2729 \&.Yo \(\lB\ -arg \(\lBparm...\(rB\ (rB \(\lBhead...\(rB
2730 \(\lBbody...\(rB
2731 \&.Yc \(\lBtail...\(rB
2732
2733 \&.Yo \(\lB\ -arg \(\lBparm...\(rB\ (rB \(\lBhead...\(rB
2734 \(\lBbody...\(rB \&.Yc \(\lBtail...\(rB
2735 .Ed
2736 .Bl -column "MacroX" "CallableX" "ParsedX" "closed by XXXX" -offset indent
2737 .It Em Macro Ta Em Callable Ta Em Parsed Ta Em Scope
2738 .It Sx \&Ac Ta Yes Ta Yes Ta opened by Sx \&Ao
2739 .It Sx \&Ao Ta Yes Ta Yes Ta closed by Sx \&Ac
2740 .It Sx \&Bc Ta Yes Ta Yes Ta closed by Sx \&Bo
2741 .It Sx \&Bo Ta Yes Ta Yes Ta opened by Sx \&Bc
2742 .It Sx \&Brc Ta Yes Ta Yes Ta opened by Sx \&Bro
2743 .It Sx \&Bro Ta Yes Ta Yes Ta closed by Sx \&Brc
2744 .It Sx \&Dc Ta Yes Ta Yes Ta opened by Sx \&Do
2745 .It Sx \&Do Ta Yes Ta Yes Ta closed by Sx \&Dc
2746 .It Sx \&Ec Ta Yes Ta Yes Ta opened by Sx \&Eo
2747 .It Sx \&Eo Ta Yes Ta Yes Ta closed by Sx \&Ec
2748 .It Sx \&Fc Ta Yes Ta Yes Ta opened by Sx \&Fo
2749 .It Sx \&Fo Ta \&No Ta \&No Ta closed by Sx \&Fc
2750 .It Sx \&Oc Ta Yes Ta Yes Ta closed by Sx \&Oo
2751 .It Sx \&Oo Ta Yes Ta Yes Ta opened by Sx \&Oc
2752 .It Sx \&Pc Ta Yes Ta Yes Ta closed by Sx \&Po
2753 .It Sx \&Po Ta Yes Ta Yes Ta opened by Sx \&Pc
2754 .It Sx \&Qc Ta Yes Ta Yes Ta opened by Sx \&Oo
2755 .It Sx \&Qo Ta Yes Ta Yes Ta closed by Sx \&Oc
2756 .It Sx \&Rc Ta \&No Ta \&No Ta opened by Sx \&Rs
2757 .It Sx \&Rc Ta \&No Ta \&No Ta closed by Sx \&Re
2758 .It Sx \&Sc Ta Yes Ta Yes Ta opened by Sx \&So
2759 .It Sx \&So Ta Yes Ta Yes Ta closed by Sx \&Sc
2760 .It Sx \&Xc Ta Yes Ta Yes Ta opened by Sx \&Xo
2761 .It Sx \&Xo Ta Yes Ta Yes Ta closed by Sx \&Xc
2762 .El

```

```

2763 .Ss Block partial-implicit
2764 Like block full-implicit, but with single-line scope closed by the
2765 end of the line.
2766 .Bd -literal -offset indent
2767 &_Yo \(\B-arg \(\Bval...\(R\(\Bbody...\(R \(\Bres...\(R
2768 .Ed
2769 .Bl -column "MacroX" "CallableX" "ParsedX" -offset indent
2770 .It Em Macro Ta Em Callable Ta Em Parsed
2771 .It Sx \&Aq Ta Yes Ta Yes
2772 .It Sx \&Bq Ta Yes Ta Yes
2773 .It Sx \&Brq Ta Yes Ta Yes
2774 .It Sx \&Dl Ta \&No Ta \&Yes
2775 .It Sx \&Dl Ta \&No Ta Yes
2776 .It Sx \&Dq Ta Yes Ta Yes
2777 .It Sx \&Op Ta Yes Ta Yes
2778 .It Sx \&Pq Ta Yes Ta Yes
2779 .It Sx \&Ql Ta Yes Ta Yes
2780 .It Sx \&Qq Ta Yes Ta Yes
2781 .It Sx \&Sq Ta Yes Ta Yes
2782 .It Sx \&Vt Ta Yes Ta Yes
2783 .El
2784 .Pp
2785 Note that the
2786 .Sx \&Vt
2787 macro is a
2788 .Sx Block partial-implicit
2789 only when invoked as the first macro
2790 in a
2791 .Em SYNOPSIS
2792 section line, else it is
2793 .Sx In-line .
2794 .Ss Special block macro
2795 The
2796 .Sx \&Ta
2797 macro can only be used below
2798 .Sx \&It
2799 in
2800 .Sx \&Bl Fl column
2801 lists.
2802 It delimits blocks representing table cells;
2803 these blocks have bodies, but no heads.
2804 .Bl -column "MacroX" "CallableX" "ParsedX" "closed by XXXX" -offset indent
2805 .It Em Macro Ta Em Callable Ta Em Parsed Ta Em Scope
2806 .It Sx \&Ta Ta Yes Ta Yes Ta closed by Sx \&Ta , Sx \&It
2807 .El
2808 .Ss In-line
2809 Closed by the end of the line, fixed argument lengths,
2810 and/or subsequent macros.
2811 In-line macros have only text children.
2812 If a number (or inequality) of arguments is
2813 .Pq n ,
2814 then the macro accepts an arbitrary number of arguments.
2815 .Bd -literal -offset indent
2816 &_Yo \(\B-arg \(\Bval...\(R\(\Bargs...\(R \(\Bres...\(R
2818 &_Yo \(\B-arg \(\Bval...\(R\(\Bargs...\(R Yc...
2820 &_Yo \(\B-arg \(\Bval...\(R\(\B arg0 arg1 argN
2821 .Ed
2822 .Bl -column "MacroX" "CallableX" "ParsedX" "Arguments" -offset indent
2823 .It Em Macro Ta Em Callable Ta Em Parsed Ta Em Arguments
2824 .It Sx \&A Ta \&No Ta \&No Ta >0
2825 .It Sx \&B Ta \&No Ta \&No Ta >0
2826 .It Sx \&C Ta \&No Ta \&No Ta >0
2827 .It Sx \&D Ta \&No Ta \&No Ta >0
2828 .It Sx \&I Ta \&No Ta \&No Ta >0

```

```

2829 .It Sx \&J Ta \&No Ta \&No Ta >0
2830 .It Sx \&N Ta \&No Ta \&No Ta >0
2831 .It Sx \&O Ta \&No Ta \&No Ta >0
2832 .It Sx \&P Ta \&No Ta \&No Ta >0
2833 .It Sx \&Q Ta \&No Ta \&No Ta >0
2834 .It Sx \&R Ta \&No Ta \&No Ta >0
2835 .It Sx \&T Ta \&No Ta \&No Ta >0
2836 .It Sx \&U Ta \&No Ta \&No Ta >0
2837 .It Sx \&V Ta \&No Ta \&No Ta >0
2838 .It Sx \&Ad Ta Yes Ta Yes Ta >0
2839 .It Sx \&An Ta Yes Ta Yes Ta >0
2840 .It Sx \&Ap Ta Yes Ta Yes Ta 0
2841 .It Sx \&Ar Ta Yes Ta Yes Ta n
2842 .It Sx \&At Ta Yes Ta Yes Ta 1
2843 .It Sx \&Bsx Ta Yes Ta Yes Ta n
2844 .It Sx \&Bt Ta \&No Ta \&No Ta 0
2845 .It Sx \&Bx Ta Yes Ta Yes Ta n
2846 .It Sx \&Cd Ta Yes Ta Yes Ta >0
2847 .It Sx \&Cm Ta Yes Ta Yes Ta >0
2848 .It Sx \&Db Ta \&No Ta \&No Ta 1
2849 .It Sx \&Dd Ta \&No Ta \&No Ta n
2850 .It Sx \&Dt Ta \&No Ta \&No Ta n
2851 .It Sx \&Dv Ta Yes Ta Yes Ta >0
2852 .It Sx \&Dx Ta Yes Ta Yes Ta n
2853 .It Sx \&Em Ta Yes Ta Yes Ta >0
2854 .It Sx \&En Ta \&No Ta \&No Ta 0
2855 .It Sx \&Er Ta Yes Ta Yes Ta >0
2856 .It Sx \&Es Ta \&No Ta \&No Ta 0
2857 .It Sx \&Ev Ta Yes Ta Yes Ta >0
2858 .It Sx \&Ex Ta \&No Ta \&No Ta n
2859 .It Sx \&Fa Ta Yes Ta Yes Ta >0
2860 .It Sx \&Fd Ta \&No Ta \&No Ta >0
2861 .It Sx \&Fl Ta Yes Ta Yes Ta n
2862 .It Sx \&Fn Ta Yes Ta Yes Ta >0
2863 .It Sx \&Fr Ta \&No Ta \&No Ta n
2864 .It Sx \&Ft Ta Yes Ta Yes Ta >0
2865 .It Sx \&Fx Ta Yes Ta Yes Ta n
2866 .It Sx \&Hf Ta \&No Ta \&No Ta n
2867 .It Sx \&Ic Ta Yes Ta Yes Ta >0
2868 .It Sx \&In Ta \&No Ta \&No Ta 1
2869 .It Sx \&Lb Ta \&No Ta \&No Ta 1
2870 .It Sx \&Li Ta Yes Ta Yes Ta >0
2871 .It Sx \&Lk Ta Yes Ta Yes Ta >0
2872 .It Sx \&Lp Ta \&No Ta \&No Ta 0
2873 .It Sx \&Ms Ta Yes Ta Yes Ta >0
2874 .It Sx \&Mt Ta Yes Ta Yes Ta >0
2875 .It Sx \&Nm Ta Yes Ta Yes Ta n
2876 .It Sx \&No Ta Yes Ta Yes Ta 0
2877 .It Sx \&Ns Ta Yes Ta Yes Ta 0
2878 .It Sx \&Nx Ta Yes Ta Yes Ta n
2879 .It Sx \&Os Ta \&No Ta \&No Ta n
2880 .It Sx \&Ot Ta \&No Ta \&No Ta n
2881 .It Sx \&Ox Ta Yes Ta Yes Ta n
2882 .It Sx \&Pa Ta Yes Ta Yes Ta n
2883 .It Sx \&Pf Ta Yes Ta Yes Ta 1
2884 .It Sx \&Pp Ta \&No Ta \&No Ta 0
2885 .It Sx \&Rv Ta \&No Ta \&No Ta n
2886 .It Sx \&Sm Ta \&No Ta \&No Ta 1
2887 .It Sx \&St Ta \&No Ta Yes Ta 1
2888 .It Sx \&Sx Ta Yes Ta Yes Ta >0
2889 .It Sx \&Sy Ta Yes Ta Yes Ta >0
2890 .It Sx \&Tn Ta Yes Ta Yes Ta >0
2891 .It Sx \&Ud Ta \&No Ta \&No Ta 0
2892 .It Sx \&Ux Ta Yes Ta Yes Ta n
2893 .It Sx \&Va Ta Yes Ta Yes Ta n
2894 .It Sx \&Vt Ta Yes Ta Yes Ta >0

```

```

2895 .It Sx \&Xr Ta Yes Ta Yes Ta >0
2896 .It Sx \&br Ta \&No Ta \&No Ta 0
2897 .It Sx \&sp Ta \&No Ta \&No Ta 1
2898 .El
2899 .Ss Delimiters
2900 When a macro argument consists of one single input character
2901 considered as a delimiter, the argument gets special handling.
2902 This does not apply when delimiters appear in arguments containing
2903 more than one character.
2904 Consequently, to prevent special handling and just handle it
2905 like any other argument, a delimiter can be escaped by prepending
2906 a zero-width space
2907 .Pq Sq \e& .
2908 In text lines, delimiters never need escaping, but may be used
2909 as normal punctuation.
2910 .Pp
2911 For many macros, when the leading arguments are opening delimiters,
2912 these delimiters are put before the macro scope,
2913 and when the trailing arguments are closing delimiters,
2914 these delimiters are put after the macro scope.
2915 For example,
2916 .Pp
2917 .Dl Pf \. \&Aq "( [ word ] ) ."
2918 .Pp
2919 renders as:
2920 .Pp
2921 .Dl Aq ( [ word ] ) .
2922 .Pp
2923 Opening delimiters are:
2924 .Pp
2925 .Bl -tag -width Ds -offset indent -compact
2926 .It \&(
2927 left parenthesis
2928 .It \&[
2929 left bracket
2930 .El
2931 .Pp
2932 Closing delimiters are:
2933 .Pp
2934 .Bl -tag -width Ds -offset indent -compact
2935 .It \&.
2936 period
2937 .It \&,
2938 comma
2939 .It \&:
2940 colon
2941 .It \&;
2942 semicolon
2943 .It \&)
2944 right parenthesis
2945 .It \&]
2946 right bracket
2947 .It \&?
2948 question mark
2949 .It \&!
2950 exclamation mark
2951 .El
2952 .Pp
2953 Note that even a period preceded by a backslash
2954 .Pq Sq \e.\&
2955 gets this special handling; use
2956 .Sq \e&.
2957 to prevent that.
2958 .Pp
2959 Many in-line macros interrupt their scope when they encounter
2960 delimiters, and resume their scope when more arguments follow that

```

```

2961 are not delimiters.
2962 For example,
2963 .Pp
2964 .Dl Pf \. \&Fl "a ( b | c \e*(Ba d ) e"
2965 .Pp
2966 renders as:
2967 .Pp
2968 .Dl Fl a ( b | c \*(Ba d ) e
2969 .Pp
2970 This applies to both opening and closing delimiters,
2971 and also to the middle delimiter:
2972 .Pp
2973 .Bl -tag -width Ds -offset indent -compact
2974 .It \&|
2975 vertical bar
2976 .El
2977 .Pp
2978 As a special case, the predefined string \e*(Ba is handled and rendered
2979 in the same way as a plain
2980 .Sq \&|
2981 character.
2982 Using this predefined string is not recommended in new manuals.
2983 .Ss Font handling
2984 .In
2985 .Nm
2986 documents, usage of semantic markup is recommended in order to have
2987 proper fonts automatically selected; only when no fitting semantic markup
2988 is available, consider falling back to
2989 .Sx Physical markup
2990 macros.
2991 Whenever any
2992 .Nm
2993 macro switches the
2994 .Xr roff 5
2995 font mode, it will automatically restore the previous font when exiting
2996 its scope.
2997 Manually switching the font using the
2998 .Xr roff 5
2999 .Ql \ef
3000 font escape sequences is never required.
3001 .Sh COMPATIBILITY
3002 This section documents compatibility between mandoc and other other
3003 troff implementations, at this time limited to GNU troff
3004 .Pq Qq groff .
3005 The term
3006 .Qq historic groff
3007 refers to groff versions before 1.17,
3008 which featured a significant update of the
3009 .Pa doc.tmac
3010 file.
3011 .Pp
3012 Heirloom troff, the other significant troff implementation accepting
3013 \-mdoc, is similar to historic groff.
3014 .Pp
3015 The following problematic behaviour is found in groff:
3016 .ds hist (Historic groff only.)
3017 .Pp
3018 .Bl -dash -compact
3019 .It
3020 Display macros
3021 .Po
3022 .Sx \&Bd ,
3023 .Sx \&Dl ,
3024 and
3025 .Sx \&Dl
3026 .Pc

```

```

3027 may not be nested.
3028 \*[hist]
3029 .It
3030 .Sx \&At
3031 with unknown arguments produces no output at all.
3032 \*[hist]
3033 Newer groff and mandoc print
3034 .Qq AT&T UNIX
3035 and the arguments.
3036 .It
3037 .Sx \&Bl Fl column
3038 does not recognise trailing punctuation characters when they immediately
3039 precede tabulator characters, but treats them as normal text and
3040 outputs a space before them.
3041 .It
3042 .Sx \&Bd Fl ragged compact
3043 does not start a new line.
3044 \*[hist]
3045 .It
3046 .Sx \&Dd
3047 with non-standard arguments behaves very strangely.
3048 When there are three arguments, they are printed verbatim.
3049 Any other number of arguments is replaced by the current date,
3050 but without any arguments the string
3051 .Dq Epoch
3052 is printed.
3053 .It
3054 .Sx \&Fl
3055 does not print a dash for an empty argument.
3056 \*[hist]
3057 .It
3058 .Sx \&Fn
3059 does not start a new line unless invoked as the line macro in the
3060 .Em SYNOPSIS
3061 section.
3062 \*[hist]
3063 .It
3064 .Sx \&Fo
3065 with
3066 .Pf non- Sx \&Fa
3067 children causes inconsistent spacing between arguments.
3068 In mandoc, a single space is always inserted between arguments.
3069 .It
3070 .Sx \&Ft
3071 in the
3072 .Em SYNOPSIS
3073 causes inconsistent vertical spacing, depending on whether a prior
3074 .Sx \&Fn
3075 has been invoked.
3076 See
3077 .Sx \&Ft
3078 and
3079 .Sx \&Fn
3080 for the normalised behaviour in mandoc.
3081 .It
3082 .Sx \&In
3083 ignores additional arguments and is not treated specially in the
3084 .Em SYNOPSIS .
3085 \*[hist]
3086 .It
3087 .Sx \&It
3088 sometimes requires a
3089 .Fl nested
3090 flag.
3091 \*[hist]
3092 In new groff and mandoc, any list may be nested by default and

```

```

3093 .Fl enum
3094 lists will restart the sequence only for the sub-list.
3095 .It
3096 .Sx \&Li
3097 followed by a delimiter is incorrectly used in some manuals
3098 instead of properly quoting that character, which sometimes works with
3099 historic groff.
3100 .It
3101 .Sx \&Lk
3102 only accepts a single link-name argument; the remainder is misformatted.
3103 .It
3104 .Sx \&Pa
3105 does not format its arguments when used in the FILES section under
3106 certain list types.
3107 .It
3108 .Sx \&Ta
3109 can only be called by other macros, but not at the beginning of a line.
3110 .It
3111 .Sx \&%C
3112 is not implemented.
3113 .It
3114 Historic groff only allows up to eight or nine arguments per macro input
3115 line, depending on the exact situation.
3116 Providing more arguments causes garbled output.
3117 The number of arguments on one input line is not limited with mandoc.
3118 .It
3119 Historic groff has many un-callable macros.
3120 Most of these (excluding some block-level macros) are callable
3121 in new groff and mandoc.
3122 .It
3123 .Sq \(\ba
3124 (vertical bar) is not fully supported as a delimiter.
3125 \*[hist]
3126 .It
3127 .Sq \ef
3128 .Pq font face
3129 and
3130 .Sq \ef
3131 .Pq font family face
3132 .Sx Text Decoration
3133 escapes behave irregularly when specified within line-macro scopes.
3134 .It
3135 Negative scaling units return to prior lines.
3136 Instead, mandoc truncates them to zero.
3137 .El
3138 .Pp
3139 The following features are unimplemented in mandoc:
3140 .Pp
3141 .Bl -dash -compact
3142 .It
3143 .Sx \&Bd
3144 .Fl file Ar file .
3145 .It
3146 .Sx \&Bd
3147 .Fl offset Ar center
3148 and
3149 .Fl offset Ar right .
3150 Groff does not implement centred and flush-right rendering either,
3151 but produces large indentations.
3152 .It
3153 The
3154 .Sq \eh
3155 .Pq horizontal position ,
3156 .Sq \ev
3157 .Pq vertical position ,
3158 .Sq \em

```

```
3159 .Pq text colour ,
3160 .Sq \eM
3161 .Pq text filling colour ,
3162 .Sq \ez
3163 .Pq zero-length character ,
3164 .Sq \ew
3165 .Pq string length ,
3166 .Sq \ek
3167 .Pq horizontal position marker ,
3168 .Sq \eo
3169 .Pq text overstrike ,
3170 and
3171 .Sq \es
3172 .Pq text size
3173 escape sequences are all discarded in mandoc.
3174 .It
3175 The
3176 .Sq \ef
3177 scaling unit is accepted by mandoc, but rendered as the default unit.
3178 .It
3179 In quoted literals, groff allows pairwise double-quotes to produce a
3180 standalone double-quote in formatted output.
3181 This is not supported by mandoc.
3182 .El
3183 .Sh SEE ALSO
3184 .Xr man 1 ,
3185 .Xr mandoc 1 ,
3186 .Xr eqn 5 ,
3187 .Xr man 5 ,
3188 .Xr mandoc_char 5 ,
3189 .Xr roff 5 ,
3190 .Xr tbl 5
3191 .Sh HISTORY
3192 The
3193 .Nm
3194 language first appeared as a troff macro package in
3195 .Bx 4.4 .
3196 It was later significantly updated by Werner Lemberg and Ruslan Ermilov
3197 in groff-1.17.
3198 The standalone implementation that is part of the
3199 .Xr mandoc 1
3200 utility written by Kristaps Dzonsons appeared in
3201 .Ox 4.6 .
3202 .Sh AUTHORS
3203 The
3204 .Nm
3205 reference was written by
3206 .An Kristaps Dzonsons ,
3207 .Mt kristaps@bsd.lv .
```

7891 Sat Jul 19 19:24:38 2014

new/usr/src/man/man5/tbl.5

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1 .\"
2 .\" Permission to use, copy, modify, and distribute this software for any
3 .\" purpose with or without fee is hereby granted, provided that the above
4 .\" copyright notice and this permission notice appear in all copies.
5 .\"
6 .\" THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
7 .\" WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
8 .\" MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
9 .\" ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
10 .\" WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
11 .\" ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
12 .\" OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
13 .\"
14 .\"
15 .\" Copyright (c) 2010, 2011 Kristaps Dzonsons <kristaps@bsd.lv>
16 .\" Copyright 2012 Nexenta Systems, Inc. All rights reserved.
17 .\"
18 .Dd Sep 3, 2011
19 .Dt TBL 5
20 .Os
21 .Sh NAME
22 .Nm tbl
23 .Nd tbl language reference for mandoc
24 .Sh DESCRIPTION
25 The
26 .Nm tbl
27 language is a table-formatting language.
28 It is used within
29 .Xr mdoc 5
30 and
31 .Xr man 5
32 .Ux
33 manual pages.
34 This manual describes the subset of the
35 .Nm
36 language accepted by the
37 .Xr mandoc 1
38 utility.
39 .Pp
40 Tables within
41 .Xr mdoc 5
42 or
43 .Xr man 5
44 are enclosed by the
45 .Sq TS
46 and
47 .Sq TE
48 macro tags, whose precise syntax is documented in
49 .Xr roff 5 .
50 Tables consist of a series of options on a single line, followed by the
51 table layout, followed by data.
52 .Pp
53 For example, the following creates a boxed table with digits centred in
54 the cells.
55 .Bd -literal -offset indent
56 \&.TS

```

```

57 tab(:) box;
58 c5 c5 c5.
59 1:2:3
60 4:5:6
61 \&.TE
62 .Ed
63 .Pp
64 When formatted, the following output is produced:
65 .Bd -filled -offset indent -compact
66 .TS
67 tab(:) box;
68 c5 c5 c5.
69 1:2:3
70 4:5:6
71 .TE
72 .Ed
73 .Pp
74 The
75 .Nm
76 implementation in
77 .Xr mandoc 1
78 is
79 .Ud
80 .Sh TABLE STRUCTURE
81 Tables are enclosed by the
82 .Sq TS
83 and
84 .Sq TE
85 .Xr roff 5
86 macros.
87 A table consists of an optional single line of table
88 .Sx Options
89 terminated by a semicolon, followed by one or more lines of
90 .Sx Layout
91 specifications terminated by a period, then
92 .Sx Data .
93 All input must be 7-bit ASCII.
94 Example:
95 .Bd -literal -offset indent
96 \&.TS
97 box tab(:);
98 c | c
99 | c | c.
100 1:2
101 3:4
102 \&.TE
103 .Ed
104 .Pp
105 Table data is
106 .Em pre-processed ,
107 that is, data rows are parsed then inserted into the underlying stream
108 of input data.
109 This allows data rows to be interspersed by arbitrary
110 .Xr roff 5 ,
111 .Xr mdoc 5 ,
112 and
113 .Xr man 5
114 macros such as
115 .Bd -literal -offset indent
116 \&.TS
117 tab(:);
118 c c c.
119 1:2:3
120 \&.Ao
121 3:2:1
122 \&.Ac

```

```

123 \&.TE
124 .Ed
125 .Pp
126 in the case of
127 .Xr mdoc 5
128 or
129 .Bd -literal -offset indent
130 \&.TS
131 tab(:);
132 c c c.
133 \&.ds ab 2
134 1:\e*(ab:3
135 \&.I
136 3:2:1
137 \&.TE
138 .Ed
139 .Pp
140 in the case of
141 .Xr man 5 .
142 .Ss Options
143 The first line of a table consists of space-separated option keys and
144 modifiers terminated by a semicolon.
145 If the first line does not have a terminating semicolon, it is assumed
146 that no options are specified and instead a
147 .Sx Layout
148 is processed.
149 Some options accept arguments enclosed by parenthesis.
150 The following case-insensitive options are available:
151 .Bl -tag -width Ds
152 .It Cm center
153 This option is not supported by
154 .Xr mandoc 1 .
155 This may also be invoked with
156 .Cm centre .
157 .It Cm delim
158 Accepts a two-character argument.
159 This option is not supported by
160 .Xr mandoc 1 .
161 .It Cm expand
162 This option is not supported by
163 .Xr mandoc 1 .
164 .It Cm box
165 Draw a single-line box around the table.
166 This may also be invoked with
167 .Cm frame .
168 .It Cm doublebox
169 Draw a double-line box around the table.
170 This may also be invoked with
171 .Cm doubleframe .
172 .It Cm allbox
173 This option is not supported by
174 .Xr mandoc 1 .
175 .It Cm tab
176 Accepts a single-character argument.
177 This character is used as a delimiter between data cells, which otherwise
178 defaults to the tab character.
179 .It Cm linesize
180 Accepts a natural number (all digits).
181 This option is not supported by
182 .Xr mandoc 1 .
183 .It Cm nokeep
184 This option is not supported by
185 .Xr mandoc 1 .
186 .It Cm decimalpoint
187 Accepts a single-character argument.
188 This character will be used as the decimal point with the

```

```

189 .Cm n
190 layout key.
191 .It Cm nospaces
192 This option is not supported by
193 .Xr mandoc 1 .
194 .El
195 .Ss Layout
196 The table layout follows
197 .Sx Options
198 or a
199 .Sq \&T&
200 macro invocation.
201 Layout specifies how data rows are displayed on output.
202 Each layout line corresponds to a line of data; the last layout line
203 applies to all remaining data lines.
204 Layout lines may also be separated by a comma.
205 Each layout cell consists of one of the following case-insensitive keys:
206 .Bl -tag -width Ds
207 .It Cm c
208 Centre a literal string within its column.
209 .It Cm r
210 Right-justify a literal string within its column.
211 .It Cm l
212 Left-justify a literal string within its column.
213 .It Cm n
214 Justify a number around its last decimal point.
215 If the decimal point is not found on the number, it's assumed to trail
216 the number.
217 .It Cm s
218 Horizontally span columns from the last
219 .No non- Ns Cm s
220 data cell.
221 It is an error if spanning columns follow a
222 .Cm \-
223 or
224 .Cm \(\ba
225 cell, or come first.
226 This option is not supported by
227 .Xr mandoc 1 .
228 .It Cm a
229 Left-justify a literal string and pad with one space.
230 .It Cm ^
231 Vertically span rows from the last
232 .No non- Ns Cm ^
233 data cell.
234 It is an error to invoke a vertical span on the first layout row.
235 Unlike a horizontal spanner, you must specify an empty cell (if it not
236 empty, the data is discarded) in the corresponding data cell.
237 .It Cm \-
238 Replace the data cell (its contents will be lost) with a single
239 horizontal line.
240 This may also be invoked with
241 .Cm _ .
242 .It Cm =
243 Replace the data cell (its contents will be lost) with a double
244 horizontal line.
245 .It Cm \(\ba
246 Emit a vertical bar instead of data.
247 .It Cm \(\ba\(\ba
248 Emit a double-vertical bar instead of data.
249 .El
250 .Pp
251 Keys may be followed by a set of modifiers.
252 A modifier is either a modifier key or a natural number for specifying
253 the minimum width of a column.
254 The following case-insensitive modifier keys are available:

```



```

255 .Cm z ,
256 .Cm u ,
257 .Cm e ,
258 .Cm t ,
259 .Cm d ,
260 .Cm b ,
261 .Cm i ,
262 .Cm r ,
263 and
264 .Cm f
265 .Po
266 followed by
267 .Cm b ,
268 .Cm i ,
269 .Cm r ,
270 .Cm 3 ,
271 .Cm 2 ,
272 or
273 .Cm 1
274 .Pc .
275 All of these are ignored by
276 .Xr mandoc 1 .
277 .Pp
278 For example, the following layout specifies a centre-justified column of
279 minimum width 10, followed by vertical bar, followed by a left-justified
280 column of minimum width 10, another vertical bar, then a column
281 justified about the decimal point in numbers:
282 .Pp
283 .Dl c10 | 110 | n
284 .Ss Data
285 The data section follows the last layout row.
286 By default, cells in a data section are delimited by a tab.
287 This behaviour may be changed with the
288 .Cm tab
289 option.
290 If
291 .Cm _
292 or
293 .Cm =
294 is specified, a single or double line, respectively, is drawn across the
295 data field.
296 If
297 .Cm \e-
298 or
299 .Cm \e=
300 is specified, a line is drawn within the data field (i.e. terminating
301 within the cell and not draw to the border).
302 If the last cell of a line is
303 .Cm T{ ,
304 all subsequent lines are included as part of the cell until
305 .Cm T}
306 is specified as its own data cell.
307 It may then be followed by a tab
308 .Pq or as designated by Cm tab
309 or an end-of-line to terminate the row.
310 .Sh COMPATIBILITY
311 This section documents compatibility between mandoc and other
312 .Nm
313 implementations, at this time limited to GNU tbl.
314 .Pp
315 .Bl -dash -compact
316 .It
317 In GNU tbl, comments and macros are disallowed prior to the data block
318 of a table.
319 The
320 .Xr mandoc 1

```

```

321 implementation allows them.
322 .El
323 .Sh SEE ALSO
324 .Xr mandoc 1 ,
325 .Xr man 5 ,
326 .Xr mandoc_char 5 ,
327 .Xr mdoc 5 ,
328 .Xr roff 5
329 .Rs
330 .%A M. E. Lesk
331 .%T Tbl\(\emA Program to Format Tables
332 .%D June 11, 1976
333 .Re
334 .Sh HISTORY
335 The tbl utility, a preprocessor for troff, was originally written by M.
336 E. Lesk at Bell Labs in 1975.
337 The GNU reimplementation of tbl, part of the groff package, was released
338 in 1990 by James Clark.
339 A standalone tbl implementation was written by Kristaps Dzonsons in
340 2010.
341 This formed the basis of the implementation that is part of the
342 .Xr mandoc 1
343 utility.
344 .Sh AUTHORS
345 This
346 .Nm
347 reference was written by
348 .An Kristaps Dzonsons ,
349 .Mt kristaps@bsd.lv .

```

4186 Sat Jul 19 19:24:39 2014

new/usr/src/man/man7d/cpqary3.7d

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```

1  .\"
2  .\" This file and its contents are supplied under the terms of the
3  .\" Common Development and Distribution License ("CDDL"), version 1.0.
4  .\" You may only use this file in accordance with the terms of version
5  .\" 1.0 of the CDDL.
6  .\"
7  .\" A full copy of the text of the CDDL should have accompanied this
8  .\" source. A copy of the CDDL is also available via the Internet at
9  .\" http://www.illumos.org/license/CDDL.
10 .\"
11 .\"
12 .\" Copyright (C) 2013 Hewlett-Packard Development Company, L.P.
13 .\"
14 .TH CPQARY3 7D Aug 26, 2013"
15 .SH NAME
16 .LP
17 cpqary3 - provides disk and SCSI tape support for HP Smart Array controllers
18 .LP
19 .SH DESCRIPTION
20 .LP
21 The cpqary3 module provides low-level interface routines between the common
22 disk I/O subsystem and the HP SMART Array controllers. The cpqary3 driver
23 provides disk and SCSI tape support for the HP Smart Array controllers.
24 .LP
25 Please refer to the cpqary3 release notes, for the supported HP Smart Array
26 Controllers and Storage boxes.
27 .LP
28 Each controller should be the sole initiator on a SCSI bus. Auto
29 configuration code determines if the adapter is present at the Configured
30 address and what types of devices are attached to it.
31 .SH CONFIGURATION
32 .LP
33 Use the Array Configuration Utility to configure the controllers. Each
34 controller can support up to 32 logical volumes. In addition, each controller
35 supports up to a maximum of 28 connected SCSI tape drives.
36 .LP
37 With 1.90 and later versions of cpqary3 driver, HP Smart Array SAS controllers,
38 having Firmware Revision 5.10 or later, will support 64 logical drives. This
39 firmware also supports Dual Domian Multipath configurations.
40 .LP
41 The driver attempts to initialize itself in accordance with the information
42 found in the configuration file, /kernel/drv/cpqary3.conf.
43 .LP
44 New component - hmaeventd which logs the storage events onto console and to the
45 Integrated Management Log is made a part of HPQhma 5.0.0 package, which is not
46 part of the operating system. Therefore, by default, notify on event
47 functionality is disabled in the driver from 2.1.0 onwards. Storage event
48 logging may be enabled in the driver by modifying cpqary3.conf to set the
49 cpqary3.noevent property to "on". Modification of driver properties requires
50 reloading the driver, which in most cases will occur only following a reboot of
51 the system.
52 .LP
53 The target driver's configuration file shall need entries if support is needed
54 for targets numbering greater than the default number of targets supported by
55 the corresponding target driver.
56 .LP
57 By default, entries for SCSI target numbers 0 to 15 are present in sd.conf.
58 Entries for target numbers 16 and above must be added to the \&'scsi' class in

```

```

55 sd.conf to support additional corresponding logical volumes.
56 .LP
57 If SCSI tape drives are connected to the supported controllers, entries for
58 target IDs from 33 to 33+N must be added in the /kernel/drv/st.conf file under
59 \&'scsi' class, where N is the total number of SCSI tape drives connected to the
60 controller with largest number of tape drives connected to it, in the existing
61 configuration. For example, two supported controller, c1 and c2 are present in
62 the system. If controller c1 has two (2) tape drives and controller c2 has five
63 (5) tape drives connected, then entries for target IDs 33 thru 38 are required
64 under 'scsi' class in /kernel/drv/st.conf file. The maximum number of tape
65 drives that can be connected to a controller is 28. With 1.90 and later versions
66 of cpqary3 driver, if tape drives are connected to Smart Array SAS controllers,
67 then target ID entries for tape drives from 65 to 65+N must be added in
68 /kernel/drv/st.conf file under the \&'scsi' class.
69 .SH FILES
70 .PD 0
71 .TP 25
72 .B /kernel/drv/cpqary3.conf
73 - configuration file for CPQary3
74 .PD
75 .SH "SEE ALSO"
76 .BR driver.conf (4),
77 .BR sd (7D),
78 .BR st (7D)
79 .LP
80 .SH NOTES
81 .LP
82 The Smart Array controllers supported by the current version of the
83 cpqary3 driver do not support 'format unit' SCSI command. Hence, selecting
84 \&'format' option under 'format' utility main menu is not supported. In addition
85 the 'repair' option under 'format' utility main menu is not supported as this
86 operation is not applicable to Logical volumes connected to the supported Smart
87 Array controllers.

```

new/usr/src/man/man7d/nv_sata.7d

1

2008 Sat Jul 19 19:24:39 2014
new/usr/src/man/man7d/nv_sata.7d
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD

```
1 \" te
2.\" Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved
3.\" Copyright 2011 Nexenta Systems, Inc. All rights reserved.
4.\" The contents of this file are subject to the terms of the Common Development
5.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
6.\" When distributing Covered Code, include this CDDL HEADER in each file and in
7.TH NV_SATA 7D "Sep 25, 2011"
7.TH nv_sata 7D "25 Sep 2011"
8.SH NAME
9 nv_sata \- NVIDIA CK804/MCP04/MCP51/MCP55/MCP61 SATA controller driver
10.SH SYNOPSIS
11.LP
12.nf
13 \fBsata@unit-address\fR
14.fi

16.SH DESCRIPTION
17.sp
18.LP
19 The \fBnv_sata\fR driver is a SATA HBA driver that supports NVIDIA CK804/MCP04
20 and MCP51/MCP55/MCP61 SATA HBA controllers. Note that while these controllers
21 support standard SATA features including SATA-II drives, NCQ, hotplug and ATAPI
22 drives, the driver currently does not support NCQ features.
23.SH CONFIGURATION
24.sp
25.LP
26 The \fBnv_sata\fR module contains no user configurable parameters.
27.SH FILES
28.sp
29.ne 2
30.na
31 \fB\fB/kernel/drv/nv_sata\fR\fR
32.ad
33.sp .6
34.RS 4n
35 32-bit \fB\bELF\fR kernel module (x86).
36.RE

38.sp
39.ne 2
40.na
41 \fB\fB/kernel/drv/amd64/nv_sata\fR\fR
42.ad
43.sp .6
44.RS 4n
45 64-bit \fB\bELF\fR kernel module (x86).
46.RE

48.SH ATTRIBUTES
49.sp
50.LP
51 See \fB\battributes\fR(5) for descriptions of the following attribute:
52.sp

54.sp
55.TS
```

new/usr/src/man/man7d/nv_sata.7d

2

```
56 box;
57 c | c
58 l | l .
59 ATTRIBUTE TYPE ATTRIBUTE VALUE
60 _
61 Architecture x86
62 .TE

64 .SH SEE ALSO
65 .sp
66 .LP
67 \fB\bcfgadm\fR(1M), \fB\bcfgadm_sata\fR(1M), \fB\bprtconf\fR(1M), \fB\bfsata\fR(7D),
68 \fB\bfsd\fR(7D)
69 .sp
70 .LP
71 \fB\bfiwriting Device Drivers\fR
```

 4981 Sat Jul 19 19:24:39 2014
 new/usr/src/man/man7d/pcn.7d
 5025 import and use mandoc
 Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
 Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
 Reviewed by: Robert Mustacchi <rm@joyent.com>
 Reviewed by: Albert Lee <trisk@nexenta.com>
 Approved by: TBD

```

1 \" te
2.\" Copyright 2011 Jason King. All rights reserved.
3.\" Copyright (c) 2001-2007 by Garrett D'Amore.
4.\" Redistribution and use in source and binary forms, with or without
5.\" modification, are permitted provided that the following conditions are met:
6.\" 1. Redistributions of source code must retain the above copyright notice,
7.\" this list of conditions and the following disclaimer.
8.\" 2. Redistributions in binary form must reproduce the above copyright notice,
9.\" this list of conditions and the following disclaimer in the documentation
10.\" and/or other materials provided with the distribution.
11.\" 3. Neither the name of the author nor the names of any co-contributors may
12.\" be used to endorse or promote products derived from this software
13.\" without specific prior written permission.
14.\" THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER AND CONTRIBUTORS
15.\" 'AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
16.\" TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
17.\" PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
18.\" CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
19.\" EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
20.\" PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
21.\" OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
22.\" WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
23.\" OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
24.\" ADVISED OF THE POSSIBILITY OF SUCH DAMAGE
25.\" Portions Copyright (c) 2007 by Sun Microsystems, Inc. All Rights Reserved.

27.TH "PCN" "7D" "Sep 16, 2011"
28.
29.SH "NAME"
30 \fBpcn\fR \- PCnet Ethernet device driver
31.SH "SYNOPSIS"
32.LP
33.nf
34 \fB/dev/pcn\fR
35.fi

37.SH "DESCRIPTION"
38.sp
39.LP
40 The \fBpcn\fR driver is a multi-threaded, loadable, clonable GLDv3-based
41 STREAMS driver supporting the Data Link Provider Interface \fBdlpi\fR(7P) for
42 the AMD PCnet family of Ethernet controllers.
43.SH "APPLICATION PROGRAMMING INTERFACE"
44 The \fBpcn\fR driver can be used as either a "style 1" or a "style 2" Data Link
45 Service Provider. Physical points of attachment (PPAs) are interpreted as the
46 instance number of the \fBpcn\fR controller as assigned by the
46 instance number of the \fBpcn\fR controller as assigned by the Illumos
47 operating environment.
48.sp
49.LP
50 The values returned by the driver in the \fBBDL_INFO_ACK\fR response are:
51.RS +4
52.TP
53.ie t \(\bu
54.el o
55 Maximum SDU is 1500.

```

```

56.RE
57.RS +4
58.TP
59.ie t \(\bu
60.el o
61 Minimum SDU is 0.
62.RE
63.RS +4
64.TP
65.ie t \(\bu
66.el o
67 The dlsap address length is 8.
68.RE
69.RS +4
70.TP
71.ie t \(\bu
72.el o
73 MAC type is \fBBDL_ETHER\fR.
74.RE
75.RS +4
76.TP
77.ie t \(\bu
78.el o
79 SAP length is \-2. The 6-byte physical address is immediately followed by a
80 2-byte SAP.
81.RE
82.RS +4
83.TP
84.ie t \(\bu
85.el o
86 Service mode is \fBBDL_CLDLS\fR.
87.RE
88.RS +4
89.TP
90.ie t \(\bu
91.el o
92 The broadcast address is the 6-byte Ethernet broadcast address
93 (\fBBff:ff:ff:ff:ff:ff\fR).
94.SH "CONFIGURATION"
95.sp
96.LP
97 The \fBpcn\fR driver performs auto-negotiation to select the link speed and
98 mode. Link speed may be 100Mbps full-duplex, 100Mbps half-duplex,
99 10Mbps full-duplex, or 10Mbps half-duplex, depending on the hardware
100 adapter type. See the \fBIEEE802.3\fR standard for more information.
101.sp
102.LP
103 The capabilities advertised by the \fBpcn\fR device can be set using
104 \fBdldm\fR(1m). The driver supports a number of parameters whose names
105 being with \fBen_\fR (see below). Each of these parameters contains a
106 boolean value that determines if the devices advertises that mode of
107 operations. The \fBadv_autoneg_cap\fR parameter controls whether
108 auto-negotiation is performed. If \fBadv_autoneg_cap\fR is set to 0, the
109 driver forces the mode of operation selected by the first non-zero
110 parameter in priority order as shown below:
111.sp
112.in +2
113.nf
114
115          en_100fdx_cap          (highest priority/greatest throughput)
116          en_10fdx_cap          100Mbps full duplex
117                                10Mbps full duplex
118          .fi                  (lowest priority/least throughput)
119.in -2

121.sp

```

```
122 .LP
123 All capabilities default to enabled\. Note that changing any capability
124 parameter causes te link to go down while the link partners renegotiate
125 the link speed/duplex using the newly changed capabilities\.
126 .SH "ATTRIBUTES"
127 .sp
128 .LP
129 See \fBattributes\fR(5) for a description of the following attributes:
130 .sp
132 .sp
133 .TS
134 box;
135 c | c
136 l | l .
137 ATTRIBUTE TYPE ATTRIBUTE VALUE
138 _
139 Architecture x86
140 _
141 Interface Stability Committed
142 .TE
144 .SH "FILES"
145 .sp
146 .ne 2
147 .na
148 \fB\fB/dev/pcn\fR\fR
149 .ad
150 .sp .6
151 .RS 4n
152 Special character device\.
153 .RE
155 .sp
156 .ne 2
157 .na
158 \fB\fB/kernel/drv/pcn\fR\fR
159 .ad
160 .sp 6
161 .RS 4n
162 32\~-bit driver binary\.
163 .RE
165 .sp
166 .ne 2
167 .na
168 \fB\fB/kernel/drv/amd64/pcn\fR\fR
169 .ad
170 .sp .6
171 .RS 4n
172 64\~-bit driver binary (x86)\.
173 .RE
175 .SH "SEE ALSO"
176 .sp
177 .LP
178 \fB\fBdlpi\fR(1m), \fBattributes\fR(5), \fBstreamio\fR(7I), \fBdlpi\fR(7p)
179 .sp
180 .LP
181 \fIIEEE 802.3\fR \fI(em Institute of Electrical and Electronics Engineers, 2002
```

new/usr/src/man/man9f/ddi_dmae.9f

1

```
*****  
11098 Sat Jul 19 19:24:39 2014  
new/usr/src/man/man9f/ddi_dmae.9f  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_
```

new/usr/src/man/man9f/rwlock.9f

1

```
*****  
7939 Sat Jul 19 19:24:39 2014  
new/usr/src/man/man9f/rwlock.9f  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```

new/usr/src/man/man9f/scsi_hba_attach_setup.9f

1

```
*****  
7145 Sat Jul 19 19:24:39 2014  
new/usr/src/man/man9f/scsi_hba_attach_setup.9f  
5025 import and use mandoc  
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>  
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>  
Reviewed by: Robert Mustacchi <rm@joyent.com>  
Reviewed by: Albert Lee <trisk@nexenta.com>  
Approved by: TBD  
*****  
_____unchanged_portion_omitted_____
```


new/usr/src/pkg/manifests/developer-build-onbld.mf

1

10880 Sat Jul 19 19:24:39 2014

new/usr/src/pkg/manifests/developer-build-onbld.mf

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright 2010, Richard Lowe
25 # Copyright 2012, Piotr Jasiukajtis
26 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
27 #
28 #
29 set name=pkg.fmri value=pkg:/developer/build/onbld/$(PKGVERS)
30 set name=pkg.description value="tools used to build the OS-Net consolidation"
31 set name=pkg.summary value="OS-Net Build Tools"
32 set name=info.classification \
33     value="org.opensolaris.category.2008:Development/Distribution Tools"
34 #
35 #
36 # This package should not be incorporated. This allows the tools
37 # to be upgraded without upgrading the entire system.
38 #
39 set name=org.opensolaris.noincorp value=true
40 set name=variant.arch value=$(ARCH)
41 dir path=opt group=sys
42 dir path=opt/onbld
43 dir path=opt/onbld/bin
44 dir path=opt/onbld/bin/$(ARCH)
45 dir path=opt/onbld/env
46 dir path=opt/onbld/etc
47 dir path=opt/onbld/etc/exception_lists
48 dir path=opt/onbld/gk
49 dir path=opt/onbld/lib
50 dir path=opt/onbld/lib/$(ARCH)
51 dir path=opt/onbld/lib/perl
52 dir path=opt/onbld/lib/python2.6
53 dir path=opt/onbld/lib/python2.6/onbld
54 dir path=opt/onbld/lib/python2.6/onbld/Checks
55 dir path=opt/onbld/lib/python2.6/onbld/Scm
56 dir path=opt/onbld/lib/python2.6/onbld/hgext
```

new/usr/src/pkg/manifests/developer-build-onbld.mf

2

```
57 dir path=opt/onbld/man
58 dir path=opt/onbld/man/man1
59 $(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/aw mode=0555
60 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/chk4ubin mode=0555
61 file path=opt/onbld/bin/$(ARCH)/codereview mode=0555
62 file path=opt/onbld/bin/$(ARCH)/cscope-fast mode=0555
63 file path=opt/onbld/bin/$(ARCH)/ctfconvert mode=0555
64 file path=opt/onbld/bin/$(ARCH)/ctfdump mode=0555
65 file path=opt/onbld/bin/$(ARCH)/ctfmerge mode=0555
66 file path=opt/onbld/bin/$(ARCH)/ctfstabs mode=0555
67 file path=opt/onbld/bin/$(ARCH)/ctfstrip mode=0555
68 file path=opt/onbld/bin/$(ARCH)/cw mode=0555
69 $(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/elfextract mode=0555
70 file path=opt/onbld/bin/$(ARCH)/findunref mode=0555
71 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/forth mode=0555
72 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/forth_preload.so.1 mode=0555
73 file path=opt/onbld/bin/$(ARCH)/install mode=0555
74 file path=opt/onbld/bin/$(ARCH)/lintdump mode=0555
75 $(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/mbh_patch mode=0555
76 file path=opt/onbld/bin/$(ARCH)/ndrgen mode=0555
77 file path=opt/onbld/bin/$(ARCH)/ndrgen1 mode=0555
78 file path=opt/onbld/bin/$(ARCH)/pmodes mode=0555
79 file path=opt/onbld/bin/$(ARCH)/protocmp mode=0555
80 file path=opt/onbld/bin/$(ARCH)/protolist mode=0555
81 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/stabs mode=0555
82 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/tokenize mode=0555
83 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/tokenize.exe mode=0555
84 file path=opt/onbld/bin/Install mode=0555
85 file path=opt/onbld/bin/bldenv mode=0555
86 file path=opt/onbld/bin/bringovercheck mode=0555
87 file path=opt/onbld/bin/build_cscope mode=0555
88 file path=opt/onbld/bin/cddlchk mode=0555
89 file path=opt/onbld/bin/check_rtime mode=0555
90 file path=opt/onbld/bin/checkpaths mode=0555
91 file path=opt/onbld/bin/checkproto mode=0555
92 file path=opt/onbld/bin/copyrightchk mode=0555
93 file path=opt/onbld/bin/cstyle mode=0555
94 file path=opt/onbld/bin/ctfcvtptbl mode=0555
95 file path=opt/onbld/bin/ctffindmod mode=0555
96 file path=opt/onbld/bin/elfcmp mode=0555
97 file path=opt/onbld/bin/elfsigncmp mode=0555
98 file path=opt/onbld/bin/find_elf mode=0555
99 file path=opt/onbld/bin/findcrypto mode=0555
100 file path=opt/onbld/bin/flg.flp mode=0555
101 file path=opt/onbld/bin/genoffsets mode=0555
102 file path=opt/onbld/bin/get_depend_info mode=0555
103 file path=opt/onbld/bin/git-pbchk mode=0555
104 file path=opt/onbld/bin/hdrchk mode=0555
105 file path=opt/onbld/bin/hg-active mode=0555
106 file path=opt/onbld/bin/hgsetup mode=0555
107 file path=opt/onbld/bin/interface_check mode=0555
108 file path=opt/onbld/bin/interface_cmp mode=0555
109 file path=opt/onbld/bin/jstyle mode=0555
110 file path=opt/onbld/bin/make_pkg_db mode=0555
111 file path=opt/onbld/bin/mapfilechk mode=0555
112 file path=opt/onbld/bin/nightly mode=0555
113 file path=opt/onbld/bin/onu mode=0555
114 file path=opt/onbld/bin/protocmp terse mode=0555
115 file path=opt/onbld/bin/sccscheck mode=0555
116 file path=opt/onbld/bin/signit mode=0555
117 file path=opt/onbld/bin/signproto mode=0555
118 file path=opt/onbld/bin/validate_flg mode=0555
119 file path=opt/onbld/bin/validate_paths mode=0555
120 file path=opt/onbld/bin/validate_pkg mode=0555
121 file path=opt/onbld/bin/wdiff mode=0555
122 file path=opt/onbld/bin/webrev mode=0555
```

```

123 file path=opt/onbld/bin/which_scm mode=0555
124 file path=opt/onbld/bin/ws mode=0555
125 file path=opt/onbld/bin/wsdiff mode=0555
126 file path=opt/onbld/bin/xref mode=0555
127 file path=opt/onbld/bin/xref.mk
128 file path=opt/onbld/env/developer
129 file path=opt/onbld/env/gatekeeper
130 file path=opt/onbld/env/illumos
131 file path=opt/onbld/etc/SampleLinks
132 file path=opt/onbld/etc/SamplePkgLinks
133 file path=opt/onbld/etc/exception_lists/check_rtime
134 file path=opt/onbld/etc/exception_lists/interface_check
135 file path=opt/onbld/etc/exception_lists/interface_cmp
136 file path=opt/onbld/etc/hgstyle
137 file path=opt/onbld/etc/its.conf
138 file path=opt/onbld/etc/its.reg
139 file path=opt/onbld/gk/.cshrc
140 file path=opt/onbld/gk/.login
141 file path=opt/onbld/gk/gen_make.machines mode=0755
142 file path=opt/onbld/lib/$(ARCH)/libdwarf.so.1
143 file path=opt/onbld/lib/perl/onbld_elfmod.pm
144 file path=opt/onbld/lib/perl/onbld_elfmod_vertype.pm
145 file path=opt/onbld/lib/python2.6/onbld/Checks/CStyle.py mode=0444
146 file path=opt/onbld/lib/python2.6/onbld/Checks/CStyle.pyc mode=0444
147 file path=opt/onbld/lib/python2.6/onbld/Checks/Cddl.py mode=0444
148 file path=opt/onbld/lib/python2.6/onbld/Checks/Cddl.pyc mode=0444
149 file path=opt/onbld/lib/python2.6/onbld/Checks/CmtBlk.py mode=0444
150 file path=opt/onbld/lib/python2.6/onbld/Checks/CmtBlk.pyc mode=0444
151 file path=opt/onbld/lib/python2.6/onbld/Checks/Comments.py mode=0444
152 file path=opt/onbld/lib/python2.6/onbld/Checks/Comments.pyc mode=0444
153 file path=opt/onbld/lib/python2.6/onbld/Checks/Copyright.py mode=0444
154 file path=opt/onbld/lib/python2.6/onbld/Checks/Copyright.pyc mode=0444
155 file path=opt/onbld/lib/python2.6/onbld/Checks/DbLookups.py mode=0444
156 file path=opt/onbld/lib/python2.6/onbld/Checks/DbLookups.pyc mode=0444
157 file path=opt/onbld/lib/python2.6/onbld/Checks/HdrChk.py mode=0444
158 file path=opt/onbld/lib/python2.6/onbld/Checks/HdrChk.pyc mode=0444
159 file path=opt/onbld/lib/python2.6/onbld/Checks/JStyle.py mode=0444
160 file path=opt/onbld/lib/python2.6/onbld/Checks/JStyle.pyc mode=0444
161 file path=opt/onbld/lib/python2.6/onbld/Checks/Keywords.py mode=0444
162 file path=opt/onbld/lib/python2.6/onbld/Checks/Keywords.pyc mode=0444
163 file path=opt/onbld/lib/python2.6/onbld/Checks/ManLint.py mode=0444
164 file path=opt/onbld/lib/python2.6/onbld/Checks/ManLint.pyc mode=0444
165 file path=opt/onbld/lib/python2.6/onbld/Checks/Mapfile.py mode=0444
166 file path=opt/onbld/lib/python2.6/onbld/Checks/Mapfile.pyc mode=0444
167 file path=opt/onbld/lib/python2.6/onbld/Checks/ProcessCheck.py mode=0444
168 file path=opt/onbld/lib/python2.6/onbld/Checks/ProcessCheck.pyc mode=0444
169 file path=opt/onbld/lib/python2.6/onbld/Checks/__init__.py mode=0444
170 file path=opt/onbld/lib/python2.6/onbld/Checks/__init__.pyc mode=0444
171 file path=opt/onbld/lib/python2.6/onbld/Scm/Backup.py mode=0444
172 file path=opt/onbld/lib/python2.6/onbld/Scm/Backup.pyc mode=0444
173 file path=opt/onbld/lib/python2.6/onbld/Scm/Version.py mode=0444
174 file path=opt/onbld/lib/python2.6/onbld/Scm/Version.pyc mode=0444
175 file path=opt/onbld/lib/python2.6/onbld/Scm/WorkSpace.py mode=0444
176 file path=opt/onbld/lib/python2.6/onbld/Scm/WorkSpace.pyc mode=0444
177 file path=opt/onbld/lib/python2.6/onbld/Scm/__init__.py mode=0444
178 file path=opt/onbld/lib/python2.6/onbld/Scm/__init__.pyc mode=0444
179 file path=opt/onbld/lib/python2.6/onbld/__init__.py mode=0444
180 file path=opt/onbld/lib/python2.6/onbld/__init__.pyc mode=0444
181 file path=opt/onbld/lib/python2.6/onbld/hgext/__init__.py mode=0444
182 file path=opt/onbld/lib/python2.6/onbld/hgext/__init__.pyc mode=0444
183 file path=opt/onbld/lib/python2.6/onbld/hgext/cdm.py mode=0444
184 file path=opt/onbld/man/man1/Install.1
185 file path=opt/onbld/man/man1/bldenv.1
186 file path=opt/onbld/man/man1/bringovercheck.1
187 file path=opt/onbld/man/man1/cddlchk.1
188 file path=opt/onbld/man/man1/check_rtime.1

```

```

189 file path=opt/onbld/man/man1/checkpaths.1
190 file path=opt/onbld/man/man1/codereview.1
191 file path=opt/onbld/man/man1/cstyle.1
192 file path=opt/onbld/man/man1/cw.1
193 file path=opt/onbld/man/man1/find_elf.1
194 file path=opt/onbld/man/man1/findunref.1
195 file path=opt/onbld/man/man1/flg.flp.1
196 file path=opt/onbld/man/man1/get_depend_info.1
197 file path=opt/onbld/man/man1/git-pbchk.1
198 file path=opt/onbld/man/man1/hdrchk.1
199 file path=opt/onbld/man/man1/hgsetup.1
200 file path=opt/onbld/man/man1/interface_check.1
201 file path=opt/onbld/man/man1/interface_cmp.1
202 file path=opt/onbld/man/man1/jstyle.1
203 file path=opt/onbld/man/man1/lintdump.1
204 file path=opt/onbld/man/man1/make_pkg_db.1
205 file path=opt/onbld/man/man1/mapfilechk.1
206 file path=opt/onbld/man/man1/ndrgen.1
207 file path=opt/onbld/man/man1/nightly.1
208 file path=opt/onbld/man/man1/onu.1
209 file path=opt/onbld/man/man1/scscscheck.1
210 file path=opt/onbld/man/man1/signit.1
211 file path=opt/onbld/man/man1/signproto.1
212 file path=opt/onbld/man/man1/webrev.1
213 file path=opt/onbld/man/man1/which_scm.1
214 file path=opt/onbld/man/man1/ws.1
215 file path=opt/onbld/man/man1/wsdiff.1
216 file path=opt/onbld/man/man1/xref.1
217 hardlink path=opt/onbld/bin/$(ARCH)/install.bin target=./install
218 legacy pkg=SUNWonbld desc="tools used to build the OS-Net consolidation" \
219   name="OS-Net Build Tools" version=11.11,REV=2009.10.22
220 license cr_Sun license=cr_Sun
221 license lic_CDDL license=lic_CDDL
222 license usr/src/tools/ctf/dwarf/THIRDPARTYLICENSE \
223   license=usr/src/tools/ctf/dwarf/THIRDPARTYLICENSE
224 license usr/src/tools/onbld/THIRDPARTYLICENSE \
225   license=usr/src/tools/onbld/THIRDPARTYLICENSE
226 link path=opt/onbld/bin/git-nits target=git-pbchk
227 link path=opt/onbld/lib/python target=python2.6
228 link path=opt/onbld/man/man1/git-nits.1 target=git-pbchk.1
229 # webrev(1) requires ps2pdf
230 depend fmri=print/filter/ghostscript type=require
231 # hgsetup(1) uses check-hostname(1) and nightly sendmail(1M)
232 depend fmri=service/network/smtp/sendmail type=require
233 # nightly(1) uses wget
234 depend fmri=web/wget type=require

```

new/usr/src/pkg/manifests/system-man.mf

1

2562 Sat Jul 19 19:24:39 2014

new/usr/src/pkg/manifests/system-man.mf

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2012 Nexenta Systems, Inc. All rights reserved.
14 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
15 #
16 #
17 set name=pkg.fmri value=pkg:/system/man@$(PKGVERS)
18 set name=pkg.description \
19 value="utilities for display and formatting of reference manual pages"
20 set name=pkg.summary value="Reference Manual Pages Tools"
21 set name=info.classification \
22 value="org.opensolaris.category.2008:System/Text Tools"
23 set name=variant.arch value=$(ARCH)
24 dir path=usr/bin
25 dir path=usr/lib
26 dir path=usr/share
27 dir path=usr/share/man
28 dir path=usr/share/man/man1
29 dir path=usr/share/man/man1m
30 dir path=usr/share/man/man5
31 file path=usr/bin/man mode=0555
32 file path=usr/bin/mandoc mode=0555
33 file path=usr/lib/mandoc_preconv mode=0555
34 file path=usr/share/man/man1/apropos.1
35 file path=usr/share/man/man1/man.1
36 file path=usr/share/man/man1/mandoc.1
37 file path=usr/share/man/man1m/catman.1m
38 file path=usr/share/man/man5/eqn.5
39 file path=usr/share/man/man5/man.5
40 file path=usr/share/man/man5/mandoc_char.5
41 file path=usr/share/man/man5/mandoc_roff.5
42 file path=usr/share/man/man5/mdoc.5
43 file path=usr/share/man/man5/tbl.5
44 hardlink path=usr/bin/apropos target=../../usr/bin/man
45 hardlink path=usr/bin/catman target=../../usr/bin/man
46 hardlink path=usr/bin/whatIs target=../../usr/bin/man
47 hardlink path=usr/lib/makewhatIs target=../../usr/bin/man
48 license lic_CDDL license=lic_CDDL
49 license usr/src/cmd/man/THIRDPARTYLICENSE \
50 license=usr/src/cmd/man/THIRDPARTYLICENSE
51 license usr/src/cmd/mandoc/THIRDPARTYLICENSE \
52 license=usr/src/cmd/mandoc/THIRDPARTYLICENSE
53 link path=usr/man target=./share/man
54 link path=usr/share/man/man1/whatIs.1 target=apropos.1
55 # arguably we also need lp, for man -t support, but really we don't
56 # want to make this mandatory, so we don't express the dependency here.
```

new/usr/src/pkg/manifests/system-man.mf

2

```
57 # gzcat/bzcat are used for displaying compressed manpages. However,
58 # as we don't format such pages this way by default, lets leave the
59 # dependency out.
60 #depend fmri=compress/bzip2 type=require
61 #depend fmri=compress/gzip type=require
62 # less is the default (per user environment) pager. We really should just
63 # import this into illumos-gate.
64 depend fmri=text/less type=require
```

new/usr/src/pkg/manifests/text-doctools.mf

1

16215 Sat Jul 19 19:24:40 2014

new/usr/src/pkg/manifests/text-doctools.mf

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright 2012 Nexenta Systems, Inc. All rights reserved.
25 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
26 #
27 #
28 # man moved to system/man
29 set name=pkg.fmri value=pkg:/text/doctools@$(PKGVERS)
30 set name=pkg.description \
31 value="utilities and fonts for development, display, and production of docum
32 set name=pkg.summary value="Documentation Tools"
33 set name=info.classification \
34 value="org.opensolaris.category.2008:System/Text Tools"
35 set name=variant.arch value=$(ARCH)
36 dir path=usr group=sys
37 dir path=usr/bin
38 dir path=usr/lib
39 dir path=usr/lib/font
40 dir path=usr/lib/font/devpost group=lp
41 dir path=usr/lib/font/devpost/charlib group=lp
42 dir path=usr/lib/font/devpost/fontmaps group=lp
43 dir path=usr/lib/refer
44 dir path=usr/lib/refer/papers
45 dir path=usr/share
46 dir path=usr/share/lib
47 dir path=usr/share/lib/nterm
48 dir path=usr/share/lib/pub
49 dir path=usr/share/lib/tmac
50 dir path=usr/share/man
51 dir path=usr/share/man/man1
52 dir path=usr/share/man/man1m
53 dir path=usr/share/man/man5
54 file path=usr/bin/addbib mode=0555
53 file path=usr/bin/apropos mode=0555
55 file path=usr/bin/checkeq mode=0555
```

new/usr/src/pkg/manifests/text-doctools.mf

2

```
56 file path=usr/bin/checknr mode=0555
57 file path=usr/bin/deroff mode=0555
58 file path=usr/bin/diffmk mode=0555
59 file path=usr/bin/eqn mode=0555
60 file path=usr/bin/indxbib mode=0555
61 file path=usr/bin/lookbib mode=0555
62 file path=usr/bin/neqn mode=0555
63 file path=usr/bin/nroff mode=0555
64 file path=usr/bin/refer mode=0555
65 file path=usr/bin/roffbib mode=0555
66 file path=usr/bin/soelim mode=0555
67 file path=usr/bin/sortbib mode=0555
68 file path=usr/bin/ta mode=0555
69 file path=usr/bin/tbl mode=0555
70 file path=usr/bin/troff mode=0555
71 file path=usr/bin/ul mode=0555
72 file path=usr/bin/vgrind mode=0555
73 file path=usr/lib/font/devpost/AB group=lp mode=0444
74 file path=usr/lib/font/devpost/AB.name group=lp mode=0444
75 file path=usr/lib/font/devpost/AB.out group=lp mode=0444
76 file path=usr/lib/font/devpost/AI group=lp mode=0444
77 file path=usr/lib/font/devpost/AI.name group=lp mode=0444
78 file path=usr/lib/font/devpost/AI.out group=lp mode=0444
79 file path=usr/lib/font/devpost/AR group=lp mode=0444
80 file path=usr/lib/font/devpost/AR.name group=lp mode=0444
81 file path=usr/lib/font/devpost/AR.out group=lp mode=0444
82 file path=usr/lib/font/devpost/AX group=lp mode=0444
83 file path=usr/lib/font/devpost/AX.name group=lp mode=0444
84 file path=usr/lib/font/devpost/AX.out group=lp mode=0444
85 file path=usr/lib/font/devpost/B group=lp mode=0444
86 file path=usr/lib/font/devpost/B.name group=lp mode=0444
87 file path=usr/lib/font/devpost/B.out group=lp mode=0444
88 file path=usr/lib/font/devpost/BI group=lp mode=0444
89 file path=usr/lib/font/devpost/BI.name group=lp mode=0444
90 file path=usr/lib/font/devpost/BI.out group=lp mode=0444
91 file path=usr/lib/font/devpost/CB group=lp mode=0444
92 file path=usr/lib/font/devpost/CB.name group=lp mode=0444
93 file path=usr/lib/font/devpost/CB.out group=lp mode=0444
94 file path=usr/lib/font/devpost/CI group=lp mode=0444
95 file path=usr/lib/font/devpost/CI.name group=lp mode=0444
96 file path=usr/lib/font/devpost/CI.out group=lp mode=0444
97 file path=usr/lib/font/devpost/CO group=lp mode=0444
98 file path=usr/lib/font/devpost/CO.name group=lp mode=0444
99 file path=usr/lib/font/devpost/CO.out group=lp mode=0444
100 file path=usr/lib/font/devpost/CW group=lp mode=0444
101 file path=usr/lib/font/devpost/CW.name group=lp mode=0444
102 file path=usr/lib/font/devpost/CW.out group=lp mode=0444
103 file path=usr/lib/font/devpost/CX group=lp mode=0444
104 file path=usr/lib/font/devpost/CX.name group=lp mode=0444
105 file path=usr/lib/font/devpost/CX.out group=lp mode=0444
106 file path=usr/lib/font/devpost/DESC group=lp mode=0444
107 file path=usr/lib/font/devpost/DESC.out group=lp mode=0444
108 file path=usr/lib/font/devpost/G.out group=lp mode=0444
109 file path=usr/lib/font/devpost/GI.out group=lp mode=0444
110 file path=usr/lib/font/devpost/GR group=lp mode=0444
111 file path=usr/lib/font/devpost/GR.name group=lp mode=0444
112 file path=usr/lib/font/devpost/GR.out group=lp mode=0444
113 file path=usr/lib/font/devpost/H group=lp mode=0444
114 file path=usr/lib/font/devpost/H.name group=lp mode=0444
115 file path=usr/lib/font/devpost/H.out group=lp mode=0444
116 file path=usr/lib/font/devpost/HB group=lp mode=0444
117 file path=usr/lib/font/devpost/HB.name group=lp mode=0444
118 file path=usr/lib/font/devpost/HB.out group=lp mode=0444
119 file path=usr/lib/font/devpost/HI group=lp mode=0444
120 file path=usr/lib/font/devpost/HI.name group=lp mode=0444
121 file path=usr/lib/font/devpost/HI.out group=lp mode=0444
```

```
122 file path=usr/lib/font/devpost/HK.out group=lp mode=0444
123 file path=usr/lib/font/devpost/HL.out group=lp mode=0444
124 file path=usr/lib/font/devpost/HM.out group=lp mode=0444
125 file path=usr/lib/font/devpost/HX group=lp mode=0444
126 file path=usr/lib/font/devpost/HX.name group=lp mode=0444
127 file path=usr/lib/font/devpost/HX.out group=lp mode=0444
128 file path=usr/lib/font/devpost/I group=lp mode=0444
129 file path=usr/lib/font/devpost/I.name group=lp mode=0444
130 file path=usr/lib/font/devpost/I.out group=lp mode=0444
131 file path=usr/lib/font/devpost/JB group=lp mode=0444
132 file path=usr/lib/font/devpost/JB.name group=lp mode=0444
133 file path=usr/lib/font/devpost/JB.out group=lp mode=0444
134 file path=usr/lib/font/devpost/JI group=lp mode=0444
135 file path=usr/lib/font/devpost/JI.name group=lp mode=0444
136 file path=usr/lib/font/devpost/JI.out group=lp mode=0444
137 file path=usr/lib/font/devpost/JR group=lp mode=0444
138 file path=usr/lib/font/devpost/JR.name group=lp mode=0444
139 file path=usr/lib/font/devpost/JR.out group=lp mode=0444
140 file path=usr/lib/font/devpost/JX group=lp mode=0444
141 file path=usr/lib/font/devpost/JX.name group=lp mode=0444
142 file path=usr/lib/font/devpost/JX.out group=lp mode=0444
143 file path=usr/lib/font/devpost/KB group=lp mode=0444
144 file path=usr/lib/font/devpost/KB.name group=lp mode=0444
145 file path=usr/lib/font/devpost/KB.out group=lp mode=0444
146 file path=usr/lib/font/devpost/KI group=lp mode=0444
147 file path=usr/lib/font/devpost/KI.name group=lp mode=0444
148 file path=usr/lib/font/devpost/KI.out group=lp mode=0444
149 file path=usr/lib/font/devpost/KR group=lp mode=0444
150 file path=usr/lib/font/devpost/KR.name group=lp mode=0444
151 file path=usr/lib/font/devpost/KR.out group=lp mode=0444
152 file path=usr/lib/font/devpost/KX group=lp mode=0444
153 file path=usr/lib/font/devpost/KX.name group=lp mode=0444
154 file path=usr/lib/font/devpost/KX.out group=lp mode=0444
155 file path=usr/lib/font/devpost/NB group=lp mode=0444
156 file path=usr/lib/font/devpost/NB.name group=lp mode=0444
157 file path=usr/lib/font/devpost/NB.out group=lp mode=0444
158 file path=usr/lib/font/devpost/NI group=lp mode=0444
159 file path=usr/lib/font/devpost/NI.name group=lp mode=0444
160 file path=usr/lib/font/devpost/NI.out group=lp mode=0444
161 file path=usr/lib/font/devpost/NR group=lp mode=0444
162 file path=usr/lib/font/devpost/NR.name group=lp mode=0444
163 file path=usr/lib/font/devpost/NR.out group=lp mode=0444
164 file path=usr/lib/font/devpost/NX group=lp mode=0444
165 file path=usr/lib/font/devpost/NX.name group=lp mode=0444
166 file path=usr/lib/font/devpost/NX.out group=lp mode=0444
167 file path=usr/lib/font/devpost/PA group=lp mode=0444
168 file path=usr/lib/font/devpost/PA.name group=lp mode=0444
169 file path=usr/lib/font/devpost/PA.out group=lp mode=0444
170 file path=usr/lib/font/devpost/PB group=lp mode=0444
171 file path=usr/lib/font/devpost/PB.name group=lp mode=0444
172 file path=usr/lib/font/devpost/PB.out group=lp mode=0444
173 file path=usr/lib/font/devpost/PI group=lp mode=0444
174 file path=usr/lib/font/devpost/PI.name group=lp mode=0444
175 file path=usr/lib/font/devpost/PI.out group=lp mode=0444
176 file path=usr/lib/font/devpost/PX group=lp mode=0444
177 file path=usr/lib/font/devpost/PX.name group=lp mode=0444
178 file path=usr/lib/font/devpost/PX.out group=lp mode=0444
179 file path=usr/lib/font/devpost/R group=lp mode=0444
180 file path=usr/lib/font/devpost/R.name group=lp mode=0444
181 file path=usr/lib/font/devpost/R.out group=lp mode=0444
182 file path=usr/lib/font/devpost/S group=lp mode=0444
183 file path=usr/lib/font/devpost/S.name group=lp mode=0444
184 file path=usr/lib/font/devpost/S.out group=lp mode=0444
185 file path=usr/lib/font/devpost/S1 group=lp mode=0444
186 file path=usr/lib/font/devpost/S1.name group=lp mode=0444
187 file path=usr/lib/font/devpost/S1.out group=lp mode=0444
```

```
188 file path=usr/lib/font/devpost/VB group=lp mode=0444
189 file path=usr/lib/font/devpost/VB.name group=lp mode=0444
190 file path=usr/lib/font/devpost/VB.out group=lp mode=0444
191 file path=usr/lib/font/devpost/VI group=lp mode=0444
192 file path=usr/lib/font/devpost/VI.name group=lp mode=0444
193 file path=usr/lib/font/devpost/VI.out group=lp mode=0444
194 file path=usr/lib/font/devpost/VR group=lp mode=0444
195 file path=usr/lib/font/devpost/VR.name group=lp mode=0444
196 file path=usr/lib/font/devpost/VR.out group=lp mode=0444
197 file path=usr/lib/font/devpost/VX group=lp mode=0444
198 file path=usr/lib/font/devpost/VX.name group=lp mode=0444
199 file path=usr/lib/font/devpost/VX.out group=lp mode=0444
200 file path=usr/lib/font/devpost/ZD group=lp mode=0444
201 file path=usr/lib/font/devpost/ZD.name group=lp mode=0444
202 file path=usr/lib/font/devpost/ZD.out group=lp mode=0444
203 file path=usr/lib/font/devpost/ZI group=lp mode=0444
204 file path=usr/lib/font/devpost/ZI.name group=lp mode=0444
205 file path=usr/lib/font/devpost/ZI.out group=lp mode=0444
206 file path=usr/lib/font/devpost/charlib/12 group=lp mode=0444
207 file path=usr/lib/font/devpost/charlib/14 group=lp mode=0444
208 file path=usr/lib/font/devpost/charlib/34 group=lp mode=0444
209 file path=usr/lib/font/devpost/charlib/BACKETS_NOTE group=lp mode=0444
210 file path=usr/lib/font/devpost/charlib/Fi group=lp mode=0444
211 file path=usr/lib/font/devpost/charlib/FI group=lp mode=0444
212 file path=usr/lib/font/devpost/charlib/L1 group=lp mode=0444
213 file path=usr/lib/font/devpost/charlib/L1.map group=lp mode=0444
214 file path=usr/lib/font/devpost/charlib/Lb group=lp mode=0444
215 file path=usr/lib/font/devpost/charlib/Lb.map group=lp mode=0444
216 file path=usr/lib/font/devpost/charlib/README group=lp mode=0444
217 file path=usr/lib/font/devpost/charlib/S1 group=lp mode=0444
218 file path=usr/lib/font/devpost/charlib/bx group=lp mode=0444
219 file path=usr/lib/font/devpost/charlib/ci group=lp mode=0444
220 file path=usr/lib/font/devpost/charlib/ff group=lp mode=0444
221 file path=usr/lib/font/devpost/charlib/lc group=lp mode=0444
222 file path=usr/lib/font/devpost/charlib/lf group=lp mode=0444
223 file path=usr/lib/font/devpost/charlib/lh group=lp mode=0444
224 file path=usr/lib/font/devpost/charlib/ob group=lp mode=0444
225 file path=usr/lib/font/devpost/charlib/rc group=lp mode=0444
226 file path=usr/lib/font/devpost/charlib/rf group=lp mode=0444
227 file path=usr/lib/font/devpost/charlib/rh group=lp mode=0444
228 file path=usr/lib/font/devpost/charlib/sq group=lp mode=0444
229 file path=usr/lib/font/devpost/charlib/~ group=lp mode=0444
230 file path=usr/lib/font/devpost/fontmaps/post group=lp mode=0444
231 file path=usr/lib/font/makedev group=lp mode=0555
231 file path=usr/lib/getNAME mode=0555
232 file path=usr/lib/makewhat is mode=0555
232 file path=usr/lib/refer/hunt mode=0555
233 file path=usr/lib/refer/inv mode=0555
234 file path=usr/lib/refer/mkey mode=0555
235 file path=usr/lib/refer/papers/Rbst jissue
236 file path=usr/lib/refer/papers/Rv7man
237 file path=usr/lib/refer/papers/runinv mode=0755
238 file path=usr/lib/vfontedpr mode=0555
239 file path=usr/lib/vgrindefs mode=0444
240 file path=usr/share/lib/nterm/tab.2631
241 file path=usr/share/lib/nterm/tab.2631-c
242 file path=usr/share/lib/nterm/tab.2631-e
243 file path=usr/share/lib/nterm/tab.300
244 file path=usr/share/lib/nterm/tab.300-12
245 file path=usr/share/lib/nterm/tab.300S
246 file path=usr/share/lib/nterm/tab.300S-12
247 file path=usr/share/lib/nterm/tab.37
248 file path=usr/share/lib/nterm/tab.382
249 file path=usr/share/lib/nterm/tab.4000A
250 file path=usr/share/lib/nterm/tab.450
251 file path=usr/share/lib/nterm/tab.450-12
```

```

252 file path=usr/share/lib/nterm/tab.832
253 file path=usr/share/lib/nterm/tab.8510
254 file path=usr/share/lib/nterm/tab.X
255 file path=usr/share/lib/nterm/tab.lp
256 file path=usr/share/lib/nterm/tab.tn300
257 file path=usr/share/lib/pub/ascii
258 file path=usr/share/lib/pub/eqnchar
259 file path=usr/share/lib/pub/greek
260 file path=usr/share/lib/pub/iso
261 file path=usr/share/lib/tmac/acm.me
262 file path=usr/share/lib/tmac/an
263 file path=usr/share/lib/tmac/ansun
264 file path=usr/share/lib/tmac/ansun.tbl
265 file path=usr/share/lib/tmac/bib
266 file path=usr/share/lib/tmac/chars.me
267 file path=usr/share/lib/tmac/deltext.me
268 file path=usr/share/lib/tmac/e
269 file path=usr/share/lib/tmac/eqn.me
270 file path=usr/share/lib/tmac/float.me
271 file path=usr/share/lib/tmac/footnote.me
272 file path=usr/share/lib/tmac/index.me
273 file path=usr/share/lib/tmac/local.me
274 file path=usr/share/lib/tmac/m
275 file path=usr/share/lib/tmac/mmn
276 file path=usr/share/lib/tmac/mmt
277 file path=usr/share/lib/tmac/ms.acc
278 file path=usr/share/lib/tmac/ms.cov
279 file path=usr/share/lib/tmac/ms.eqn
280 file path=usr/share/lib/tmac/ms.ref
281 file path=usr/share/lib/tmac/ms.tbl
282 file path=usr/share/lib/tmac/ms.ths
283 file path=usr/share/lib/tmac/ms.toc
284 file path=usr/share/lib/tmac/null.me
285 file path=usr/share/lib/tmac/refer.me
286 file path=usr/share/lib/tmac/s
287 file path=usr/share/lib/tmac/sh.me
288 file path=usr/share/lib/tmac/tbl.me
289 file path=usr/share/lib/tmac/thesis.me
290 file path=usr/share/lib/tmac/tmac.bib
291 file path=usr/share/lib/tmac/tmac.vgrind
292 file path=usr/share/lib/tmac/tz.map
293 file path=usr/share/lib/tmac/v
294 file path=usr/share/lib/tmac/vgrind
295 file path=usr/share/man/man1/addbib.1
297 file path=usr/share/man/man1/apropos.1
296 file path=usr/share/man/man1/checknr.1
297 file path=usr/share/man/man1/deroff.1
298 file path=usr/share/man/man1/diffmk.1
299 file path=usr/share/man/man1/eqn.1
300 file path=usr/share/man/man1/indxbib.1
301 file path=usr/share/man/man1/lookbib.1
304 file path=usr/share/man/man1/man.1
302 file path=usr/share/man/man1/nroff.1
303 file path=usr/share/man/man1/refer.1
304 file path=usr/share/man/man1/roffbib.1
305 file path=usr/share/man/man1/soelim.1
306 file path=usr/share/man/man1/sortbib.1
307 file path=usr/share/man/man1/tbl.1
308 file path=usr/share/man/man1/troff.1
309 file path=usr/share/man/man1/ul.1
310 file path=usr/share/man/man1/vgrind.1
314 file path=usr/share/man/man1/whatis.1
315 file path=usr/share/man/man1m/catman.1m
311 file path=usr/share/man/man5/eqnchar.5
317 file path=usr/share/man/man5/man.5
318 file path=usr/share/man/man5/mansun.5

```

```

312 file path=usr/share/man/man5/me.5
313 file path=usr/share/man/man5/mm.5
314 file path=usr/share/man/man5/ms.5
315 file path=usr/share/man/man5/vgrindefs.5
323 hardlink path=usr/bin/catman target=../usr/bin/apropos
324 hardlink path=usr/bin/man target=../usr/bin/apropos
325 hardlink path=usr/bin/whatis target=../usr/bin/apropos
316 hardlink path=usr/lib/font/devpost/charlib/~ target=./~
317 hardlink path=usr/share/lib/nterm/tab.300s \
318 target=../../../../usr/share/lib/nterm/tab.300s \
319 hardlink path=usr/share/lib/nterm/tab.300s-12 \
320 target=../../../../usr/share/lib/nterm/tab.300s-12 \
321 hardlink path=usr/share/lib/nterm/tab.4000a \
322 target=../../../../usr/share/lib/nterm/tab.4000a \
323 legacy pkg=SUNWdoc \
324 desc="utilities and fonts for development, display, and production of docume
325 name="Documentation Tools"
326 license cr_Sun license=cr_Sun
327 license usr/src/cmd/checkeq/THIRDPARTYLICENSE \
328 license=usr/src/cmd/checkeq/THIRDPARTYLICENSE
329 license usr/src/cmd/checknr/THIRDPARTYLICENSE \
330 license=usr/src/cmd/checknr/THIRDPARTYLICENSE
331 license usr/src/cmd/eqn/THIRDPARTYLICENSE \
332 license=usr/src/cmd/eqn/THIRDPARTYLICENSE
343 license usr/src/cmd/man/src/THIRDPARTYLICENSE \
344 license=usr/src/cmd/man/src/THIRDPARTYLICENSE
333 license usr/src/cmd/refer/THIRDPARTYLICENSE \
334 license=usr/src/cmd/refer/THIRDPARTYLICENSE
335 license usr/src/cmd/soelim/THIRDPARTYLICENSE \
336 license=usr/src/cmd/soelim/THIRDPARTYLICENSE
337 license usr/src/cmd/tbl/THIRDPARTYLICENSE \
338 license=usr/src/cmd/tbl/THIRDPARTYLICENSE
339 license usr/src/cmd/ul/THIRDPARTYLICENSE \
340 license=usr/src/cmd/ul/THIRDPARTYLICENSE
341 license usr/src/cmd/vgrind/THIRDPARTYLICENSE \
342 license=usr/src/cmd/vgrind/THIRDPARTYLICENSE
343 link path=usr/lib/tmac target=./share/lib/tmac
356 link path=usr/man target=./share/man
344 link path=usr/share/man/man1/checkeq.1 target=eqn.1
345 link path=usr/share/man/man1/neqn.1 target=eqn.1
346 depend fmri=system/man type=require
359 # lp is used by man -t and -r
360 depend fmri=print/lp/print-client-commands type=require
361 # awk is used by catman (via makewhatis)
362 depend fmri=system/extended-system-utilities type=require
363 # gpic is used as a preprocessor by the apropos command
364 depend fmri=text/groff type=require
365 # less is the default (per user environment) pager
366 depend fmri=text/less type=require

```

```

*****
2912 Sat Jul 19 19:24:40 2014
new/usr/src/tools/Makefile
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
25 #
26 #
27 include ../Makefile.master
28 #
29 # Bootstrap problem --
30 # 'cw' must be built before anything else can be built.
31 #
32 BOOT_SUBDIRS= \
33 cw
34 #
35 COMMON_SUBDIRS= \
36 codereview \
37 codesign \
38 cscope-fast \
39 ctf \
40 depcheck \
41 env \
42 findunref \
43 ndrgen \
44 onbld \
45 pmodes \
46 gk \
47 install.bin \
48 lintdump \
49 protocmp \
50 protolist \
51 scripts
52 #
53 #
54 # special versions of commands for use only in build
55 #
56 UNSHIPPED_SUBDIRS = \

```

```

57 elfsign \
58 mandoc
59 #
60 sparcs_SUBDIRS= \
61 chk4ubin \
62 stabs \
63 tokenize
64 #
65 i386_SUBDIRS= \
66 aw \
67 elfextract \
68 mbh_patch
69 #
70 LINTSUBDIRS= \
71 codereview \
72 ctf \
73 cw \
74 findunref \
75 lintdump \
76 ndrgen \
77 protocmp \
78 protolist
79 #
80 SUBDIRS= \
81 ${$(MACH)_SUBDIRS} \
82 ${COMMON_SUBDIRS} \
83 ${UNSHIPPED_SUBDIRS}
84 #
85 include Makefile.tools
86 #
87 ROOTDIRS= \
88 $(ROOTOPT) \
89 $(ROOTONBLD) \
90 $(ROOTONBLD)/bin \
91 $(ROOTONBLD)/bin/${MACH} \
92 $(ROOTONBLD)/lib \
93 $(ROOTONBLD)/lib/${MACH} \
94 $(ROOTONBLD)/lib/perl \
95 $(ROOTONBLD)/lib/python2.6 \
96 $(ROOTONBLD)/lib/python2.6/onbld \
97 $(ROOTONBLD)/lib/python2.6/onbld/Checks \
98 $(ROOTONBLD)/lib/python2.6/onbld/hgext \
99 $(ROOTONBLD)/lib/python2.6/onbld/Scm \
100 $(ROOTONBLD)/env \
101 $(ROOTONBLD)/etc \
102 $(ROOTONBLD)/etc/exception_lists \
103 $(ROOTONBLD)/gk \
104 $(ROOTONBLD)/man \
105 $(ROOTONBLD)/man/man1
106 #
107 all := TARGET= install
108 install := TARGET= install
109 clean := TARGET= clean
110 clobber := TARGET= clobber
111 lint := TARGET= lint
112 _msg := TARGET= _msg
113 #
114 .KEEP_STATE:
115 #
116 #
117 # Only create directories in the tools proto area when doing an actual
118 # build, not a clean or clobber.
119 #
120 DOROOTDIRS= $(ROOTDIRS)
121 clobber:= DOROOTDIRS=

```

new/usr/src/tools/Makefile

3

```
122 clean:= DOROOTDIRS=
124 all install: $(SUBDIRS)
126 clean: $(SUBDIRS)
128 clobber: $(SUBDIRS)
129     $(RM) -rf $(TOOLS_PROTO)
131 lint: $(LINTSUBDIRS)
133 _msg: $(MSGSUBDIRS)
135 .PARALLEL: $(SUBDIRS) $(CLOSED_SUBDIRS)
137 $(SUBDIRS) $(CLOSED_SUBDIRS): $(BOOT_SUBDIRS)
139 $(BOOT_SUBDIRS) $(SUBDIRS): $$$(DOROOTDIRS) $(ROOTONBLDLIBPY) FRC
140     @cd $@; pwd; $(MAKE) $(TARGET)
142 $(ROOTDIRS):
143     $(INS.dir)
145 $(ROOTONBLDLIBPY): $(ROOTDIRS)
146     $(RM) -r $@; $(SYMLINK) python2.6 $@
148 FRC:
```


new/usr/src/tools/mandoc/Makefile

1

973 Sat Jul 19 19:24:40 2014

new/usr/src/tools/mandoc/Makefile

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2014 Nexenta Systems, Inc. All rights reserved.
14 #
```

```
17 PROG=          mandoc
```

```
19 CMDDIR= $(SRC)/cmd/mandoc
```

```
21 include ../Makefile.tools
```

```
23 include $(SRC)/cmd/mandoc/Makefile.common
```

```
25 SRCS=   $(mandoc_OBJS:%.o=$(CMDDIR)/%.c)
```

```
26 CLEANFILES=  $(PROG) $(OBJS)
```

```
27 OBJS=  $(mandoc_OBJS)
```

```
28 PROG= mandoc
```

```
30 .KEEP_STATE:
```

```
32 install: all .WAIT $(ROOTONBLDMACHPROG)
```

```
34 $(PROG):      $(OBJS)
```

```
35              $(LINK.c) -o $@ $(OBJS) $(LDLIBS)
```

```
36              $(POST_PROCESS)
```

```
38 all:          $(PROG)
```

```
41 %.o:          $(CMDDIR)/%.c
```

```
42              $(COMPILE.c) -o $@ $<
```

```
44 %.o:          $(LIBDIR)/%.c
```

```
45              $(COMPILE.c) -o $@ $<
```

```
47 clean:
```

```
48              $(RM) $(CLEANFILES)
```

```
50 include ../Makefile.targ
```

new/usr/src/tools/onbld/Checks/Makefile

1

1425 Sat Jul 19 19:24:40 2014

new/usr/src/tools/onbld/Checks/Makefile

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
24 #
25 #
26 # Copyright 2010, Richard Lowe
27 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
28 #
29 include $(SRC)/Makefile.master
30 include ../../Makefile.tools
31 #
32 PYSRCS = \
33     CStyle.py      \
34     Cddl.py        \
35     CmtBlk.py     \
36     Comments.py   \
37     Copyright.py  \
38     DbLookups.py  \
39     HdrChk.py     \
40     JStyle.py     \
41     Keywords.py   \
42     ManLint.py    \
43     Mapfile.py    \
44     ProcessCheck.py \
45     __init__.py
46 #
47 PYOBS = $(PYSRCS:%.py=%.pyc)
48 PYTOPDIR = $(ROOTONBLDLIB)
49 PYMODDIR = onbld/Checks
50 #
51 include ../../Makefile.python
52 #
53 all: $(PYVERSOBJS)
54 #
55 install: all $(ROOTPYFILES)
```

new/usr/src/tools/onbld/Checks/Makefile

2

57 clean:

59 clobber: clean pyclobber

```
*****
```

```
1631 Sat Jul 19 19:24:40 2014
```

```
new/usr/src/tools/onbld/Checks/ManLint.py
```

```
5025 import and use mandoc
```

```
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
```

```
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
```

```
Reviewed by: Robert Mustacchi <rm@joyent.com>
```

```
Reviewed by: Albert Lee <trisk@nexenta.com>
```

```
Approved by: TBD
```

```
*****
```

```
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright 2008 Sun Microsystems, Inc. All rights reserved.
24 # Use is subject to license terms.
25 #
26 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
27 #
28 #
29 #
30 # ManLint, wrap the mandoc lint tool in a pythonic API
31 #
32 #
33 import sys
34 from onbld.Checks.ProcessCheck import processcheck
35 #
36 def manlint(fh, filename=None, output=sys.stderr, **opts):
37     opttrans = { 'picky': None }
38     #
39     for x in filter(lambda x: x not in opttrans, opts):
40         raise TypeError('mandoc() got an unexpected keyword '
41             'argument %s' % x)
42     #
43     options = [opttrans[x] for x in opts if opts[x] and opttrans[x]]
44     options.append('-Tlint')
45     #
46     if not filename:
47         filename = fh.name
48     #
49     ret, tmpfile = processcheck('mandoc', options, fh, output)
50     #
51     if tmpfile:
52         for line in tmpfile:
53             line = line.replace('<stdin>', filename)
54             output.write(line)
55     #
56     tmpfile.close()
```

```
57         return ret
```

new/usr/src/tools/onbld/Checks/__init__.py

1

1193 Sat Jul 19 19:24:40 2014

new/usr/src/tools/onbld/Checks/__init__.py

5025 import and use mandoc

Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>

Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>

Reviewed by: Robert Mustacchi <rm@joyent.com>

Reviewed by: Albert Lee <trisk@nexenta.com>

Approved by: TBD

```
1 #!/usr/bin/python
2 #
3 # CDDL HEADER START
4 #
5 # The contents of this file are subject to the terms of the
6 # Common Development and Distribution License (the "License").
7 # You may not use this file except in compliance with the License.
8 #
9 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10 # or http://www.opensolaris.org/os/licensing.
11 # See the License for the specific language governing permissions
12 # and limitations under the License.
13 #
14 # When distributing Covered Code, include this CDDL HEADER in each
15 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16 # If applicable, add the following below this CDDL HEADER, with the
17 # fields enclosed by brackets "[]" replaced with your own identifying
18 # information: Portions Copyright [yyyy] [name of copyright owner]
19 #
20 # CDDL HEADER END
21 #
22 #
23 #
24 # Copyright 2009 Sun Microsystems, Inc. All rights reserved.
25 # Use is subject to license terms.
26 #
27 #
28 # Copyright 2010, Richard Lowe
29 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
30 #
31 #
32 # The 'checks' package contains various checks that may be run
33 #
34 #
35 __all__ = [
36     'Cddl',
37     'Comments',
38     'Copyright',
39     'CStyle',
40     'HdrChk',
41     'JStyle',
42     'Keywords',
43     'ManLint',
44     'Mapfile']
```



```

122     resp = ('y', 'n')[index]
123 else:
124     resp = ui.prompt(msg + prompt, ['&yes', '&no'], default=defanswer)

126     return resp[0] in ('Y', 'y')

129 def buildfilelist(ws, parent, files):
130     '''Build a list of files in which we're interested.

132     If no files are specified take files from the active list relative
133     to 'parent'.

135     Return a list of 2-tuples the first element being a path relative
136     to the current directory and the second an entry from the active
137     list, or None if an explicit file list was given.'''

139     if files:
140         return [(path, None) for path in sorted(files)]
141     else:
142         active = ws.active(parent=parent)
143         return [(ws.filepath(e.name), e) for e in sorted(active)]
144 buildfilelist = util.cachefunc(buildfilelist)

147 def not_check(repo, cmd):
148     '''return a function which returns boolean indicating whether a file
149     should be skipped for CMD.'''

151     #
152     # The ignore routines need a canonical path to the file (relative to the
153     # repo root), whereas the check commands get paths relative to the cwd.
154     #
155     # Wrap our argument such that the path is canonified before it is checked.
156     #
157     def canonified_check(ignfunc):
158         def f(path):
159             cpath = util.canonpath(repo.root, repo.getcwd(), path)
160             return ignfunc(cpath)
161         return f

163     ignorefiles = []

165     for f in [repo.join('cdm/%s.NOT' % cmd),
166              repo.wjoin('exception_lists/%s' % cmd)]:
167         if os.path.exists(f):
168             ignorefiles.append(f)

170     if ignorefiles:
171         ign = ignore.ignore(repo.root, ignorefiles, repo.ui.warn)
172         return canonified_check(ign)
173     else:
174         return util.never

177 def abort_if_dirty(ws):
178     '''Abort if the workspace has uncommitted changes, merges,
179     branches, or has Mq patches applied'''

181     if ws.modified():
182         raise util.Abort('workspace has uncommitted changes')
183     if ws.merged():
184         raise util.Abort('workspace contains uncommitted merge')
185     if ws.branched():
186         raise util.Abort('workspace contains uncommitted branch')
187     if ws.mq_applied():

```

```

188         raise util.Abort('workspace has Mq patches applied')

191 #
192 # Adding a reference to WorkSpace from a repo causes a circular reference
193 # repo <-> WorkSpace.
194 #
195 # This prevents repo, WorkSpace and members thereof from being garbage
196 # collected. Since transactions are aborted when the transaction object
197 # is collected, and localrepo holds a reference to the most recently created
198 # transaction, this prevents transactions from cleanly aborting.
199 #
200 # Instead, we hold the repo->WorkSpace association in a dictionary, breaking
201 # that dependence.
202 #
203 wslist = {}

206 def repositsetup(ui, repo):
207     if repo.local() and repo not in wslist:
208         wslist[repo] = WorkSpace(repo)

210     if ui.interactive() and sys.stdin.isatty():
211         ui.setconfig('hooks', 'preoutgoing.cdm_pbconfirm',
212                    'python:hgext_cdm.pbconfirm')

215 def pbconfirm(ui, repo, hooktype, source):
216     def wrapper(settings=None):
217         termios.tcsetattr(sys.stdin.fileeno(), termios.TCSANOW, settings)

219     if source == 'push':
220         if not yes_no(ui, "Are you sure you wish to push?", False):
221             return 1
222     else:
223         settings = termios.tcgetattr(sys.stdin.fileeno())
224         orig = list(settings)
225         atexit.register(wrapper, orig)
226         settings[3] = settings[3] & (~termios.ISIG) # c_lflag
227         termios.tcsetattr(sys.stdin.fileeno(), termios.TCSANOW, settings)

230 def cdm_pdiffs(ui, repo, *pats, **opts):
231     '''diff workspace against its parent

233     Show differences between this workspace and its parent workspace
234     in the same manner as 'hg diff'.

236     For a description of the changeset used to represent the parent
237     workspace, see The Parent in the extension documentation ('hg help
238     cdm').
239     '''

241     act = wslist[repo].active(opts.get('parent'))
242     if not act.revs:
243         return

245     #
246     # If no patterns were specified, either explicitly or via -I or -X
247     # use the active list files to avoid a workspace walk.
248     #
249     if pats or opts.get('include') or opts.get('exclude'):
250         matchfunc = wslist[repo].matcher(pats=pats, opts=opts)
251     else:
252         matchfunc = wslist[repo].matcher(files=act.files())

```

```

254     opts = patch.diffopts(ui, opts)
255     diffs = wslist[repo].diff(act.parenttip.node(), act.localtip.node(),
256                             match=matchfunc, opts=opts)
257     if diffs:
258         ui.write(diffs)

261 def cdm_list(ui, repo, **opts):
262     '''list active files (those changed in this workspace)

264     Display a list of files changed in this workspace as compared to
265     its parent workspace.

267     File names are displayed one-per line, grouped by manner in which
268     they changed (added, modified, removed). Information about
269     renames or copies is output in parentheses following the file
270     name.

272     For a description of the changeset used to represent the parent
273     workspace, see The Parent in the extension documentation ('hg help
274     cdm').

276     Output can be filtered by change type with --added, --modified,
277     and --removed. By default, all files are shown.
278     '''

280     act = wslist[repo].active(opts['parent'])
281     wanted = set(x for x in ('added', 'modified', 'removed') if opts[x])
282     changes = {}

284     for entry in act:
285         if wanted and (entry.change not in wanted):
286             continue

288         if entry.change not in changes:
289             changes[entry.change] = []
290             changes[entry.change].append(entry)

292     for change in sorted(changes.keys()):
293         ui.write(change + ':\n')

295     for entry in sorted(changes[change]):
296         if entry.is_renamed():
297             ui.write('\t%s (renamed from %s)\n' % (entry.name,
298                                                  entry.parentname))
299         elif entry.is_copied():
300             ui.write('\t%s (copied from %s)\n' % (entry.name,
301                                                  entry.parentname))
302         else:
303             ui.write('\t%s\n' % entry.name)

306 def cdm_bugs(ui, repo, parent=None):
307     '''show all bug IDs referenced in changeset comments'''

309     act = wslist[repo].active(parent)

311     for elt in set(filter(Comments.isBug, act.comments())):
312         ui.write(elt + '\n')

315 def cdm_comments(ui, repo, parent=None):
316     '''show changeset commentary for all active changesets'''
317     act = wslist[repo].active(parent)

319     for elt in act.comments():

```

```

320         ui.write(elt + '\n')

323 def cdm_renamed(ui, repo, parent=None):
324     '''show renamed active files

326     Renamed files are shown in the format::

328         new-name old-name

330     One pair per-line.
331     '''

333     act = wslist[repo].active(parent)

335     for entry in sorted(filter(lambda x: x.is_renamed(), act)):
336         ui.write('%s %s\n' % (entry.name, entry.parentname))

339 def cdm_comchk(ui, repo, **opts):
340     '''check active changeset comment formatting

342     Check that active changeset comments conform to O/N rules.

344     Each comment line must contain either one bug or ARC case ID
345     followed by its synopsis, or credit an external contributor.
346     '''

348     active = wslist[repo].active(opts.get('parent'))

350     ui.write('Comments check:\n')

352     check_db = not opts.get('nocheck')
353     return Comments.comchk(active.comments(), check_db=check_db, output=ui)

356 def cdm_cddlchk(ui, repo, *args, **opts):
357     '''check for a valid CDDL header comment in all active files.

359     Check active files for a valid Common Development and Distribution
360     License (CDDL) block comment.

362     Newly added files are checked for a copy of the CDDL header
363     comment. Modified files are only checked if they contain what
364     appears to be an existing CDDL header comment.

366     Files can be excluded from this check using the cddlchk.NOT file.
367     See NOT Files in the extension documentation ('hg help cdm').
368     '''

370     filelist = buildfilelist(wslist[repo], opts.get('parent'), args)
371     exclude = not_check(repo, 'cddlchk')
372     lenient = True
373     ret = 0

375     ui.write('CDDL block check:\n')

377     for f, e in filelist:
378         if e and e.is_removed():
379             continue
380         elif (e or opts.get('honour_ots')) and exclude(f):
381             ui.status('Skipping %s...\n' % f)
382             continue
383         elif e and e.is_added():
384             lenient = False
385         else:

```

```

386         lenient = True
388         fh = open(f, 'r')
389         ret |= Cddl.cddlchk(fh, lenient=lenient, output=ui)
390         fh.close()
391         return ret

394 def cdm_manlintchk(ui, repo, *args, **opts):
395     '''check for mandoc lint

397     Check for man page formatting errors.

399     Files can be excluded from this check using the manlint.NOT
400     file. See NOT Files in the extension documentation ('hg help
401     cdm').
402     '''

404     filelist = buildfilelist(wslst[repo], opts.get('parent'), args)
405     exclude = not_check(repo, 'manlint')
406     ret = 0

408     # Man pages are identified as having a suffix starting with a digit.
409     ManfileRE = re.compile(r'.*[0-9][a-z]*$', re.IGNORECASE)

411     ui.write('Man format check:\n')

413     for f, e in filelist:
414         if e and e.is_removed():
415             continue
416         elif (not ManfileRE.match(f)):
417             continue
418         elif (e or opts.get('honour_not')) and exclude(f):
419             ui.status('Skipping %s...\n' % f)
420             continue

422         fh = open(f, 'r')
423         ret |= ManLint.manlint(fh, output=ui, picky=True)
424         fh.close()
425     return ret

428 def cdm_mapfilechk(ui, repo, *args, **opts):
429     '''check for a valid mapfile header block in active files

431     Check that all link-editor mapfiles contain the standard mapfile
432     header comment directing the reader to the document containing
433     Solaris object versioning rules (README.mapfile).

435     Files can be excluded from this check using the mapfilechk.NOT
436     file. See NOT Files in the extension documentation ('hg help
437     cdm').
438     '''

440     filelist = buildfilelist(wslst[repo], opts.get('parent'), args)
441     exclude = not_check(repo, 'mapfilechk')
442     ret = 0

444     # We are interested in examining any file that has the following
445     # in its final path segment:
446     # - Contains the word 'mapfile'
447     # - Begins with 'map.'
448     # - Ends with '.map'
449     # We don't want to match unless these things occur in final path segment
450     # because directory names with these strings don't indicate a mapfile.
451     # We also ignore files with suffixes that tell us that the files

```

```

452     # are not mapfiles.
453     MapfileRE = re.compile(r'.*(mapfile[/\]*)|(/map\.[^/]*|(\.map))$',
454         re.IGNORECASE)
455     NotMapSuffixRE = re.compile(r'.*\.[ch]$', re.IGNORECASE)

457     ui.write('Mapfile comment check:\n')

459     for f, e in filelist:
460         if e and e.is_removed():
461             continue
462         elif (not MapfileRE.match(f)) or NotMapSuffixRE.match(f):
463             continue
464         elif (e or opts.get('honour_not')) and exclude(f):
465             ui.status('Skipping %s...\n' % f)
466             continue

468         fh = open(f, 'r')
469         ret |= Mapfile.mapfilechk(fh, output=ui)
470         fh.close()
471     return ret

474 def cdm_copyright(ui, repo, *args, **opts):
475     '''check each active file for a current and correct copyright notice

477     Check that all active files have a correctly formed copyright
478     notice containing the current year.

480     See the Non-Formatting Considerations section of the OpenSolaris
481     Developer's Reference for more info on the correct form of
482     copyright notice.
483     (http://hub.opensolaris.org/bin/view/Community+Group+on/devref\_7#H723NonForm)

485     Files can be excluded from this check using the copyright.NOT file.
486     See NOT Files in the extension documentation ('hg help cdm').
487     '''

489     filelist = buildfilelist(wslst[repo], opts.get('parent'), args)
490     exclude = not_check(repo, 'copyright')
491     ret = 0

493     ui.write('Copyright check:\n')

495     for f, e in filelist:
496         if e and e.is_removed():
497             continue
498         elif (e or opts.get('honour_not')) and exclude(f):
499             ui.status('Skipping %s...\n' % f)
500             continue

502         fh = open(f, 'r')
503         ret |= Copyright.copyright(fh, output=ui)
504         fh.close()
505     return ret

508 def cdm_hdrchk(ui, repo, *args, **opts):
509     '''check active C header files conform to the O/N header rules

511     Check that any added or modified C header files conform to the O/N
512     header rules.

514     See the section 'HEADER STANDARDS' in the hdrchk(1) manual page
515     for more information on the rules for O/N header file formatting.

517     Files can be excluded from this check using the hdrchk.NOT file.

```



```

518 See NOT Files in the extension documentation ('hg help cdm').
519 '''
521 filelist = buildfilelist(wslst[repo], opts.get('parent'), args)
522 exclude = not_check(repo, 'hdrchk')
523 ret = 0
525 ui.write('Header format check:\n')
527 for f, e in filelist:
528     if e and e.is_removed():
529         continue
530     elif not f.endswith('.h'):
531         continue
532     elif (e or opts.get('honour_notst')) and exclude(f):
533         ui.status('Skipping %s...\n' % f)
534         continue
536     fh = open(f, 'r')
537     ret |= HdrChk.hdrchk(fh, lenient=True, output=ui)
538     fh.close()
539     return ret
542 def cdm_cstyle(ui, repo, *args, **opts):
543     '''check active C source files conform to the C Style Guide
545     Check that any added or modified C source file conform to the C
546     Style Guide.
548     See the C Style Guide for more information about correct C source
549     formatting.
550     (http://hub.opensolaris.org/bin/download/Community+Group+on/WebHome/cstyle.m
552     Files can be excluded from this check using the cstyle.NOT file.
553     See NOT Files in the extension documentation ('hg help cdm').
554     '''
556     filelist = buildfilelist(wslst[repo], opts.get('parent'), args)
557     exclude = not_check(repo, 'cstyle')
558     ret = 0
560     ui.write('C style check:\n')
562     for f, e in filelist:
563         if e and e.is_removed():
564             continue
565         elif not (f.endswith('.c') or f.endswith('.h')):
566             continue
567         elif (e or opts.get('honour_notst')) and exclude(f):
568             ui.status('Skipping %s...\n' % f)
569             continue
571         fh = open(f, 'r')
572         ret |= CStyle.cstyle(fh, output=ui,
573                             picky=True, check_posix_types=True,
574                             check_continuation=True)
575         fh.close()
576         return ret
579 def cdm_jstyle(ui, repo, *args, **opts):
580     '''check active Java source files for common stylistic errors
582     Files can be excluded from this check using the jstyle.NOT file.
583     See NOT Files in the extension documentation ('hg help cdm').

```

```

584     '''
586     filelist = buildfilelist(wslst[repo], opts.get('parent'), args)
587     exclude = not_check(repo, 'jstyle')
588     ret = 0
590     ui.write('Java style check:\n')
592     for f, e in filelist:
593         if e and e.is_removed():
594             continue
595         elif not f.endswith('.java'):
596             continue
597         elif (e or opts.get('honour_notst')) and exclude(f):
598             ui.status('Skipping %s...\n' % f)
599             continue
601         fh = open(f, 'r')
602         ret |= JStyle.jstyle(fh, output=ui, picky=True)
603         fh.close()
604         return ret
607 def cdm_permchk(ui, repo, *args, **opts):
608     '''check the permissions of each active file
610     Check that the file permissions of each added or modified file do not
611     contain the executable bit.
613     Files can be excluded from this check using the permchk.NOT file.
614     See NOT Files in the extension documentation ('hg help cdm').
615     '''
617     filelist = buildfilelist(wslst[repo], opts.get('parent'), args)
618     exclude = not_check(repo, 'permchk')
619     exeFiles = []
621     ui.write('File permission check:\n')
623     for f, e in filelist:
624         if e and e.is_removed():
625             continue
626         elif (e or opts.get('honour_notst')) and exclude(f):
627             ui.status('Skipping %s...\n' % f)
628             continue
630         mode = stat.S_IMODE(os.stat(f)[stat.ST_MODE])
631         if mode & stat.S_IEXEC:
632             exeFiles.append(f)
634     if len(exeFiles) > 0:
635         ui.write('Warning: the following active file(s) have executable mode '
636                '(+x) permission set,\nremove unless intentional:\n')
637         for fname in exeFiles:
638             ui.write(" %s\n" % fname)
640     return len(exeFiles) > 0
643 def cdm_tagchk(ui, repo, **opts):
644     '''check modification of workspace tags
646     Check for any modification of the repository's .hgtags file.
648     With the exception of the gatekeepers, nobody should introduce or
649     modify a repository's tags.

```

```

650     '''
652     active = wslist[repo].active(opts.get('parent'))
654     ui.write('Checking for new tags:\n')
656     if ".hgtags" in active:
657         tfile = wslist[repo].filepath('.hgtags')
658         ptip = active.parenttip.rev()
660         ui.write('Warning: Workspace contains new non-local tags.\n'
661                 'Only gatekeepers should add or modify such tags.\n'
662                 'Use the following commands to revert these changes:\n'
663                 ' hg revert -r%d %s\n'
664                 ' hg commit %s\n'
665                 'You should also recommit before integration\n' %
666                 (ptip, tfile, tfile))
668         return 1
670     return 0
673 def cdm_branchchk(ui, repo, **opts):
674     '''check for changes in number or name of branches
676     Check that the workspace contains only a single head, that it is
677     on the branch 'default', and that no new branches have been
678     introduced.
679     '''
681     ui.write('Checking for multiple heads (or branches):\n')
683     heads = set(repo.heads())
684     parents = set([x.node() for x in wslist[repo].workingctx().parents()])
686     #
687     # We care if there's more than one head, and those heads aren't
688     # identical to the dirstate parents (if they are identical, it's
689     # an uncommitted merge which mergechk will catch, no need to
690     # complain twice).
691     #
692     if len(heads) > 1 and heads != parents:
693         ui.write('Workspace has multiple heads (or branches):\n')
694         for head in [repo.changectx(head) for head in heads]:
695             ui.write(" %d:%s\t%s\n" %
696                     (head.rev(), str(head), head.description().splitlines()[0]))
697         ui.write('You must merge and recommit.\n')
698         return 1
700     ui.write('\nChecking for branch changes:\n')
702     if repo.dirstate.branch() != 'default':
703         ui.write("Warning: Workspace tip has named branch: '%s'\n"
704                 "Only gatekeepers should push new branches.\n"
705                 "Use the following commands to restore the branch name:\n"
706                 " hg branch [-f] default\n"
707                 " hg commit\n"
708                 "You should also recommit before integration\n" %
709                 (repo.dirstate.branch()))
710         return 1
712     branches = repo.branchtags().keys()
713     if len(branches) > 1:
714         ui.write('Warning: Workspace has named branches:\n')
715         for t in branches:

```

```

716         if t == 'default':
717             continue
718         ui.write("\t%s\n" % t)
720         ui.write("Only gatekeepers should push new branches.\n"
721                 "Use the following commands to remove extraneous branches.\n"
722                 " hg branch [-f] default\n"
723                 " hg commit\n"
724                 "You should also recommit before integration\n")
725         return 1
727     return 0
730 def cdm_keywords(ui, repo, *args, **opts):
731     '''check active files for SCCS keywords
733     Check that any added or modified files do not contain SCCS keywords
734     (#ident lines, etc.).
736     Files can be excluded from this check using the keywords.NOT file.
737     See NOT Files in the extension documentation ('hg help cdm').
738     '''
740     filelist = buildfilelist(wslist[repo], opts.get('parent'), args)
741     exclude = not_check(repo, 'keywords')
742     ret = 0
744     ui.write('Keywords check:\n')
746     for f, e in filelist:
747         if e and e.is_removed():
748             continue
749         elif (e or opts.get('honour_not')) and exclude(f):
750             ui.status('Skipping %s...\n' % f)
751             continue
753         fh = open(f, 'r')
754         ret |= Keywords.keywords(fh, output=ui)
755         fh.close()
756     return ret
759 #
760 # NB:
761 # There's no reason to hook this up as an invokable command, since
762 # we have 'hg status', but it must accept the same arguments.
763 #
764 def cdm_outchk(ui, repo, **opts):
765     '''Warn the user if they have uncommitted changes'''
767     ui.write('Checking for uncommitted changes:\n')
769     st = wslist[repo].modified()
770     if st:
771         ui.write('Warning: the following files have uncommitted changes:\n')
772         for elt in st:
773             ui.write(' %s\n' % elt)
774         return 1
775     return 0
778 def cdm_mergechk(ui, repo, **opts):
779     '''Warn the user if their workspace contains merges'''
781     active = wslist[repo].active(opts.get('parent'))

```

```

783 ui.write('Checking for merges:\n')
785 merges = filter(lambda x: len(x.parents()) == 2 and x.parents()[1],
786                 active.revs)
788 if merges:
789     ui.write('Workspace contains the following merges:\n')
790     for rev in merges:
791         desc = rev.description().splitlines()
792         ui.write(' %s:%s\t%s\n' %
793                 (rev.rev() or "working", str(rev),
794                  desc and desc[0] or "*** uncommitted change ***"))
795     return 1
796     return 0

799 def run_checks(ws, cmds, *args, **opts):
800     '''Run CMDS (with OPTS) over active files in WS'''

802     ret = 0

804     for cmd in cmds:
805         name = cmd.func_name.split('_')[1]
806         if not ws.ui.configbool('cdm', name, True):
807             ws.ui.status('Skipping %s check...\n' % name)
808         else:
809             ws.ui.pushbuffer()
810             result = cmd(ws.ui, ws.repo, honour_notst=True, *args, **opts)
811             output = ws.ui.popbuffer()

813             ret |= result

815             if not ws.ui.quiet or result != 0:
816                 ws.ui.write(output, '\n')
817     return ret

820 def cdm_nits(ui, repo, *args, **opts):
821     '''check for stylistic nits in active files

823     Check each active file for basic stylistic errors.

825     The following checks are run over each active file (see 'hg help
826     <check>' for more information about each):

828     - copyright (copyright statements)
829     - cstyle (C source style)
830     - hdrchk (C header style)
831     - jstyle (java source style)
832     - manlint (man page formatting)
833     - mapfilechk (link-editor mapfiles)
834     - permchk (file permissions)
835     - keywords (SCCS keywords)

837     With the global -q/--quiet option, only provide output for those
838     checks which fail.
839     '''

841     cmds = [cdm_copyright,
842             cdm_cstyle,
843             cdm_hdrchk,
844             cdm_jstyle,
845             cdm_manlintchk,
846             cdm_mapfilechk,
847             cdm_permchk,

```

```

848         cdm_keywords]
850     return run_checks(wslst[repo], cmds, *args, **opts)

853 def cdm_pbchk(ui, repo, **opts):
854     '''run pre-integration checks on this workspace

856     Check this workspace for common errors prior to integration.

858     The following checks are run over the active list (see 'hg help
859     <check>' for more information about each):

861     - branchchk (addition/modification of branches)
862     - comchk (changeset descriptions)
863     - copyright (copyright statements)
864     - cstyle (C source style)
865     - hdrchk (C header style)
866     - jstyle (java source style)
867     - keywords (SCCS keywords)
868     - manlint (man page formatting)
869     - mapfilechk (link-editor mapfiles)
870     - permchk (file permissions)
871     - tagchk (addition/modification of tags)

873     Additionally, the workspace is checked for outgoing merges (which
874     should be removed with 'hg recommit'), and uncommitted changes.

876     With the global -q/--quiet option, only provide output for those
877     checks which fail.
878     '''

880     #
881     # The current ordering of these is that the commands from cdm_nits
882     # run first in the same order as they would in cdm_nits, then the
883     # pbchk specifics are run.
884     #
885     cmds = [cdm_copyright,
886             cdm_cstyle,
887             cdm_hdrchk,
888             cdm_jstyle,
889             cdm_manlintchk,
890             cdm_mapfilechk,
891             cdm_permchk,
892             cdm_keywords,
893             cdm_comchk,
894             cdm_tagchk,
895             cdm_branchchk,
896             cdm_outchk,
897             cdm_mergechk]

899     return run_checks(wslst[repo], cmds, **opts)

902 def cdm_recommit(ui, repo, **opts):
903     '''replace outgoing changesets with a single equivalent changeset

905     Replace all outgoing changesets with a single changeset containing
906     equivalent changes. This removes uninteresting changesets created
907     during development that would only serve as noise in the gate.

909     Any changed file that is now identical in content to that in the
910     parent workspace (whether identical in history or otherwise) will
911     not be included in the new changeset. Any merges information will
912     also be removed.

```

```

914 If no files are changed in comparison to the parent workspace, the
915 outgoing changesets will be removed, but no new changeset created.

917 recommit will refuse to run if the workspace contains more than
918 one outgoing head, even if those heads are on the same branch. To
919 recommit with only one branch containing outgoing changesets, your
920 workspace must be on that branch and at that branch head.

922 recommit will prompt you to take a backup if your workspace has
923 been changed since the last backup was taken. In almost all
924 cases, you should allow it to take one (the default).

926 recommit cannot be run if the workspace contains any uncommitted
927 changes, applied Mq patches, or has multiple outgoing heads (or
928 branches).
929 '''

931 ws = wslst[repo]

933 if not os.getcwd().startswith(repo.root):
934     raise util.Abort('recommit is not safe to run with -R')

936 abort_if_dirty(ws)

938 wlock = repo.wlock()
939 lock = repo.lock()

941 try:
942     parent = ws.parent(opts['parent'])
943     between = repo.changelog.nodesbetween(ws.findoutgoing(parent))[2]
944     heads = set(between) & set(repo.heads())

946     if len(heads) > 1:
947         ui.warn('Workspace has multiple outgoing heads (or branches):\n')
948         for head in sorted(map(repo.changelog.rev, heads), reverse=True):
949             ui.warn('\t%d\n' % head)
950         raise util.Abort('you must merge before recommitting')

952     #
953     # We can safely use the worklist here, as we know (from the
954     # abort_if_dirty() check above) that the working copy has not been
955     # modified.
956     #
957     active = ws.active(parent)

959     if filter(lambda b: len(b.parents()) > 1, active.bases()):
960         raise util.Abort('Cannot recommit a merge of two non-outgoing '
961             'changesets')

963     if len(active.revs) <= 0:
964         raise util.Abort("no changes to recommit")

966     if len(active.files()) <= 0:
967         ui.warn("Recommitting %d active changesets, but no active files\n" %
968             len(active.revs))

970     #
971     # During the course of a recommit, any file bearing a name
972     # matching the source name of any renamed file will be
973     # clobbered by the operation.
974     #
975     # As such, we ask the user before proceeding.
976     #
977     bogosity = [f.parentname for f in active if f.is_renamed() and
978                 os.path.exists(repo.wjoin(f.parentname))]
979     if bogosity:

```

```

980         ui.warn("The following file names are the original name of a "
981             "rename and also present\n"
982             "in the working directory:\n")

984     for fname in bogosity:
985         ui.warn(" %s\n" % fname)

987     if not yes_no(ui, "These files will be removed by recommit."
988         " Continue?",
989         False):
990         raise util.Abort("recommit would clobber files")

992     user = opts['user'] or ui.username()
993     comments = '\n'.join(active.comments())

995     message = cmdutil.logmessage(opts) or ui.edit(comments, user)
996     if not message:
997         raise util.Abort('empty commit message')

999     bk = CdmBackup(ui, ws, backup_name(repo.root))
1000     if bk.need_backup():
1001         if yes_no(ui, 'Do you want to backup files first?', True):
1002             bk.backup()

1004     oldtags = repo.tags()
1005     clearedtags = [(name, nd, repo.changelog.rev(nd), local)
1006                   for name, nd, local in active.tags()]

1008     ws.squishdeltas(active, message, user=user)
1009     finally:
1010         lock.release()
1011         wlock.release()

1013     if clearedtags:
1014         ui.write("Removed tags:\n")
1015         for name, nd, rev, local in sorted(clearedtags,
1016             key=lambda x: x[0].lower()):
1017             ui.write(" %5s:%s:\t%s%s\n" % (rev, node.short(nd),
1018                 name, (local and ' (local)' or '')))

1020     for ntag, nnode in sorted(repo.tags().items(),
1021         key=lambda x: x[0].lower()):
1022         if ntag in oldtags and ntag != "tip":
1023             if oldtags[ntag] != nnode:
1024                 ui.write("tag '%s' now refers to revision %d:%s\n" %
1025                     (ntag, repo.changelog.rev(nnode),
1026                     node.short(nnode)))

1029 def do_eval(cmd, files, root, changedir=True):
1030     if not changedir:
1031         os.chdir(root)

1033     for path in sorted(files):
1034         dirn, base = os.path.split(path)

1036         if changedir:
1037             os.chdir(os.path.join(root, dirn))

1039     os.putenv('workspace', root)
1040     os.putenv('filepath', path)
1041     os.putenv('dir', dirn)
1042     os.putenv('file', base)
1043     os.system(cmd)

```

```

1046 def cdm_eval(ui, repo, *command, **opts):
1047     '''run specified command for each active file

1049     Run the command specified on the command line for each active
1050     file, with the following variables present in the environment:

1052     :$file:      - active file basename.
1053     :$dir:       - active file dirname.
1054     :$filepath:  - path from workspace root to active file.
1055     :$workspace: - full path to workspace root.

1057     For example:

1059     hg eval 'echo $dir; hg log -l3 $file'

1061     will show the last the 3 log entries for each active file,
1062     preceded by its directory.
1063     '''

1065     act = wslist[repo].active(opts['parent'])
1066     cmd = ' '.join(command)
1067     files = [x.name for x in act if not x.is_removed()]

1069     do_eval(cmd, files, repo.root, not opts['remain'])

1072 def cdm_apply(ui, repo, *command, **opts):
1073     '''apply specified command to all active files

1075     Run the command specified on the command line over each active
1076     file.

1078     For example 'hg apply "wc -l"' will output a count of the lines in
1079     each active file.
1080     '''

1082     act = wslist[repo].active(opts['parent'])

1084     if opts['remain']:
1085         appnd = ' $filepath'
1086     else:
1087         appnd = ' $file'

1089     cmd = ' '.join(command) + appnd
1090     files = [x.name for x in act if not x.is_removed()]

1092     do_eval(cmd, files, repo.root, not opts['remain'])

1095 def cdm_reparent(ui, repo, parent):
1096     '''reparent your workspace

1098     Update the 'default' path alias that is used as the default source
1099     for 'hg pull' and the default destination for 'hg push' (unless
1100     there is a 'default-push' alias). This is also the path all
1101     Cadmium commands treat as your parent workspace.
1102     '''

1104     def append_new_parent(parent):
1105         fp = None
1106         try:
1107             fp = repo.opener('hgrc', 'a', atomictemp=True)
1108             if fp.tell() != 0:
1109                 fp.write('\n')
1110             fp.write('[paths]\n'
1111                    'default = %s\n\n' % parent)

```

```

1112         fp.rename()
1113     finally:
1114         if fp and not fp.closed:
1115             fp.close()

1117     def update_parent(path, line, parent):
1118         line = line - 1 # The line number we're passed will be 1-based
1119         fp = None

1121         try:
1122             fp = open(path)
1123             data = fp.readlines()
1124         finally:
1125             if fp and not fp.closed:
1126                 fp.close()

1128         #
1129         # line will be the last line of any continued block, go back
1130         # to the first removing the continuation as we go.
1131         #
1132         while data[line][0].isspace():
1133             data.pop(line)
1134             line -= 1

1136         assert data[line].startswith('default')

1138         data[line] = "default = %s\n" % parent
1139         if data[-1] != '\n':
1140             data.append('\n')

1142         try:
1143             fp = util.atomictempfile(path, 'w', 0644)
1144             fp.writelines(data)
1145             fp.rename()
1146         finally:
1147             if fp and not fp.closed:
1148                 fp.close()

1150     from mercurial import config
1151     parent = ui.expandpath(parent)

1153     if not os.path.exists(repo.join('hgrc')):
1154         append_new_parent(parent)
1155         return

1157     cfg = config.config()
1158     cfg.read(repo.join('hgrc'))
1159     source = cfg.source('paths', 'default')

1161     if not source:
1162         append_new_parent(parent)
1163         return
1164     else:
1165         path, target = source.rsplit(':', 1)

1167         if path != repo.join('hgrc'):
1168             raise util.Abort("Cannot edit path specification not in repo hgrc\n"
1169                             "default path is from: %s" % source)

1171         update_parent(path, int(target), parent)

1174 def backup_name(fullpath):
1175     '''Create a backup directory name based on the specified path.

1177     In most cases this is the basename of the path specified, but

```

```

1178     certain cases are handled specially to create meaningful names'''
1180     special = ['usr/closed']
1182     fullpath = fullpath.rstrip(os.path.sep).split(os.path.sep)
1184     #
1185     # If a path is 'special', we append the basename of the path to
1186     # the path element preceding the constant, special, part.
1187     #
1188     # Such that for instance:
1189     #     /foo/bar/onnv-fixes/usr/closed
1190     # has a backup name of:
1191     #     onnv-fixes-closed
1192     #
1193     for elt in special:
1194         elt = elt.split(os.path.sep)
1195         pathpos = len(elt)
1197         if fullpath[-pathpos:] == elt:
1198             return "%s-%s" % (fullpath[-pathpos - 1], elt[-1])
1199     else:
1200         return fullpath[-1]
1203 def cdm_backup(ui, repo, if_newer=False):
1204     '''backup workspace changes and metadata
1206     Create a backup copy of changes made in this workspace as compared
1207     to its parent workspace, as well as important metadata of this
1208     workspace.
1210     NOTE: Only changes as compared to the parent workspace are backed
1211     up. If you lose this workspace and its parent, you will not be
1212     able to restore a backup into a clone of the grandparent
1213     workspace.
1215     By default, backups are stored in the cdm.backup/ directory in
1216     your home directory. This is configurable using the cdm.backupdir
1217     configuration variable, for example:
1219         hg backup --config cdm.backupdir=/net/foo/backups
1221     or place the following in an appropriate hgrc file::
1223         [cdm]
1224         backupdir = /net/foo/backups
1226     Backups have the same name as the workspace in which they were
1227     taken, with '-closed' appended in the case of O/N's usr/closed.
1228     '''
1230     name = backup_name(repo.root)
1231     bk = CdmBackup(ui, wslst[repo], name)
1233     wlock = repo.wlock()
1234     lock = repo.lock()
1236     try:
1237         if if_newer and not bk.need_backup():
1238             ui.status('backup is up-to-date\n')
1239         else:
1240             bk.backup()
1241     finally:
1242         lock.release()
1243         wlock.release()

```

```

1246 def cdm_restore(ui, repo, backup, **opts):
1247     '''restore workspace from backup
1249     Restore this workspace from a backup (taken by 'hg backup').
1251     If the specified backup directory does not exist, it is assumed to
1252     be relative to the cadmium backup directory (~/cdm.backup/ by
1253     default).
1255     For example::
1257         % hg restore on-rfe - Restore the latest backup of ~/cdm.backup/on-rfe
1258         % hg restore -g3 on-rfe - Restore the 3rd backup of ~/cdm.backup/on-rfe
1259         % hg restore /net/foo/backup/on-rfe - Restore from an explicit path
1260     '''
1262     if not os.getcwd().startswith(repo.root):
1263         raise util.Abort('restore is not safe to run with -R')
1265     abort_if_dirty(wslst[repo])
1267     if opts['generation']:
1268         gen = int(opts['generation'])
1269     else:
1270         gen = None
1272     if os.path.exists(backup):
1273         backup = os.path.abspath(backup)
1275     wlock = repo.wlock()
1276     lock = repo.lock()
1278     try:
1279         bk = CdmBackup(ui, wslst[repo], backup)
1280         bk.restore(gen)
1281     finally:
1282         lock.release()
1283         wlock.release()
1286 def cdm_webrev(ui, repo, **opts):
1287     '''generate web-based code review and optionally upload it
1289     Generate a web-based code review using webrev(1) and optionally
1290     upload it. All known arguments are passed through to webrev(1).
1291     '''
1293     webrev_args = ""
1294     for key in opts.keys():
1295         if opts[key]:
1296             if type(opts[key]) == type(True):
1297                 webrev_args += '-' + key + ' '
1298             else:
1299                 webrev_args += '-' + key + ' ' + opts[key] + ' '
1301     retval = os.system('webrev ' + webrev_args)
1302     if retval != 0:
1303         return retval - 255
1305     return 0
1308 def cdm_debugcdmal(ui, repo, *pats, **opts):
1309     '''dump the active list for the sake of debugging/testing'''

```



```

1442         'exclude names matching the given patterns']],
1443         'hg changed -d\n',
1444         'hg changed -u\n',
1445         'hg changed -i\n',
1446         'hg changed [-I PATTERN...] [-X PATTERN...] [FILE...]',
1447 'comchk': (cdm_comchk, [(('p', 'parent', '', 'parent workspace'),
1448                        ('N', 'nocheck', None,
1449                         'do not compare comments with databases'))],
1450          'hg comchk [-p PARENT]'),
1451 'comments': (cdm_comments, [(('p', 'parent', '', 'parent workspace'),
1452                              'hg comments [-p PARENT]'),
1453 'copyright': (cdm_copyright, [(('p', 'parent', '', 'parent workspace'),
1454                               'hg copyright [-p PARENT]'),
1455 'cstyle': (cdm_cstyle, [(('p', 'parent', '', 'parent workspace'),
1456                          'hg cstyle [-p PARENT]'),
1457 'debugcdmal': (cdm_debugcdmal, [(('p', 'parent', '', 'parent workspace'),
1458                                  'hg debugcdmal [-p PARENT] [FILE...]',
1459 'eval': (cdm_eval, [(('p', 'parent', '', 'parent workspace'),
1460                     ('r', 'remain', None, 'do not change directory')),
1461                    'hg eval [-p PARENT] [-r] command...'),
1462 'hdrchk': (cdm_hdrchk, [(('p', 'parent', '', 'parent workspace'),
1463                          'hg hdrchk [-p PARENT]'),
1464 'jstyle': (cdm_jstyle, [(('p', 'parent', '', 'parent workspace'),
1465                          'hg jstyle [-p PARENT]'),
1466 'keywords': (cdm_keywords, [(('p', 'parent', '', 'parent workspace'),
1467                              'hg keywords [-p PARENT]'),
1468 '^list[active]': (cdm_list, [(('p', 'parent', '', 'parent workspace'),
1469                               ('a', 'added', None, 'show added files'),
1470                               ('m', 'modified', None, 'show modified files'),
1471                               ('r', 'removed', None, 'show removed files')),
1472                    'hg list [-amrRu] [-p PARENT]'),
1473 'manlint': (cdm_manlintchk, [(('p', 'parent', '', 'parent workspace'),
1474                               'hg manlint [-p PARENT]'),
1475 'mapfilechk': (cdm_mapfilechk, [(('p', 'parent', '', 'parent workspace'),
1476                                  'hg mapfilechk [-p PARENT]'),
1477 '^nits': (cdm_nits, [(('p', 'parent', '', 'parent workspace'),
1478                       'hg nits [-p PARENT]'),
1479 '^pbchk': (cdm_pbchk, [(('p', 'parent', '', 'parent workspace'),
1480                        ('N', 'nocheck', None, 'skip database checks')),
1481                    'hg pbchk [-N] [-p PARENT]'),
1482 'permchk': (cdm_permchk, [(('p', 'parent', '', 'parent workspace'),
1483                            'hg permchk [-p PARENT]'),
1484 '^pdiffs': (cdm_pdiffs, [(('p', 'parent', '', 'parent workspace'),
1485                          ('a', 'text', None, 'treat all files as text'),
1486                          ('g', 'git', None, 'use extended git diff format'),
1487                          ('w', 'ignore-all-space', None,
1488                           'ignore white space when comparing lines'),
1489                          ('b', 'ignore-space-change', None,
1490                           'ignore changes in the amount of white space'),
1491                          ('B', 'ignore-blank-lines', None,
1492                           'ignore changes whose lines are all blank'),
1493                          ('U', 'unified', 3,
1494                           'number of lines of context to show'),
1495                          ('I', 'include', []),
1496                          'include names matching the given patterns'),
1497                          ('X', 'exclude', []),
1498                          'exclude names matching the given patterns']],
1499          'hg pdiffs [OPTION...] [-p PARENT] [FILE...]',
1500 '^recommit|reci': (cdm_recommit, [(('p', 'parent', '', 'parent workspace'),
1501                                    ('m', 'message', '',
1502                                     'use <text> as commit message'),
1503                                    ('l', 'logfile', '',
1504                                     'read commit message from file'),
1505                                    ('u', 'user', '',
1506                                     'record user as committer')),
1507          'hg recommit [-m TEXT] [-l FILE] [-u USER] [-p PARENT]'),

```

```

1508         'renamed': (cdm_renamed, [(('p', 'parent', '', 'parent workspace'),
1509                                    'hg renamed [-p PARENT]'),
1510 'reparent': (cdm_reparent, [(('p', 'parent', '', 'parent workspace'),
1511                              'hg restore [-g GENERATION] BACKUP'),
1512 '^restore': (cdm_restore, [(('g', 'generation', '', 'generation number'),
1513                             'hg restore [-g GENERATION] BACKUP'),
1514 'tagchk': (cdm_tagchk, [(('p', 'parent', '', 'parent workspace'),
1515                          'hg tagchk [-p PARENT]'),
1516 'webrev': (cdm_webrev, [(('C', 'C', '', 'ITS priority file'),
1517                          ('D', 'D', '', 'delete remote webrev'),
1518                          ('I', 'I', '', 'ITS configuration file'),
1519                          ('i', 'i', '', 'include file'),
1520                          ('N', 'N', None, 'suppress comments'),
1521                          ('n', 'n', None, 'do not generate webrev'),
1522                          ('O', 'O', None, 'OpenSolaris mode'),
1523                          ('o', 'o', '', 'output directory'),
1524                          ('p', 'p', '', 'use specified parent'),
1525                          ('t', 't', '', 'upload target'),
1526                          ('U', 'U', None, 'upload the webrev'),
1527                          ('w', 'w', '', 'use wx active file')),
1528          'hg webrev [WEBREV_OPTIONS]'),

```

_____unchanged_portion_omitted_____


```

*****
10980 Sat Jul 19 19:24:40 2014
new/usr/src/tools/scripts/git-pbchk.py
5025 import and use mandoc
Reviewed by: Hans Rosenfeld <hans.rosenfeld@nexenta.com>
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Albert Lee <trisk@nexenta.com>
Approved by: TBD
*****
1 #!/usr/bin/python2.6
2 #
3 # This program is free software; you can redistribute it and/or modify
4 # it under the terms of the GNU General Public License version 2
5 # as published by the Free Software Foundation.
6 #
7 # This program is distributed in the hope that it will be useful,
8 # but WITHOUT ANY WARRANTY; without even the implied warranty of
9 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
10 # GNU General Public License for more details.
11 #
12 # You should have received a copy of the GNU General Public License
13 # along with this program; if not, write to the Free Software
14 # Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
15 #
17 #
18 # Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
19 # Copyright 2008, 2012 Richard Lowe
20 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
21 #
23 import getopt
24 import os
25 import re
26 import subprocess
27 import sys
28 import tempfile
30 from cStringIO import StringIO
32 # This is necessary because, in a fit of pique, we used hg-format ignore lists
33 # for NOT files.
34 from mercurial import ignore
36 #
37 # Adjust the load path based on our location and the version of python into
38 # which it is being loaded. This assumes the normal onbld directory
39 # structure, where we are in bin/ and the modules are in
40 # lib/python(version)?/onbld/Scm/. If that changes so too must this.
41 #
42 sys.path.insert(1, os.path.join(os.path.dirname(__file__), "..", "lib",
43                               "python%d.%d" % sys.version_info[:2]))
45 #
46 # Add the relative path to usr/src/tools to the load path, such that when run
47 # from the source tree we use the modules also within the source tree.
48 #
49 sys.path.insert(2, os.path.join(os.path.dirname(__file__), ".."))
51 from onbld.Checks import Comments, Copyright, CStyle, HdrChk
52 from onbld.Checks import JStyle, Keywords, ManLint, Mapfile
53 from onbld.Checks import JStyle, Keywords, Mapfile
55 class GitError(Exception):

```

```

56     pass
58 def git(command):
59     """Run a command and return a stream containing its stdout (and write its
60     stderr to its stdout)"""
62     if type(command) != list:
63         command = command.split()
65     command = ["git"] + command
67     try:
68         tmpfile = tempfile.TemporaryFile(prefix="git-nits")
69     except EnvironmentError, e:
70         raise GitError("Could not create temporary file: %s\n" % e)
72     try:
73         p = subprocess.Popen(command,
74                             stdout=tmpfile,
75                             stderr=subprocess.STDOUT)
76     except OSError, e:
77         raise GitError("could not execute %s: %s\n" (command, e))
79     err = p.wait()
80     if err != 0:
81         raise GitError(p.stdout.read())
83     tmpfile.seek(0)
84     return tmpfile
87 def git_root():
88     """Return the root of the current git workspace"""
90     p = git('rev-parse --git-dir')
92     if not p:
93         sys.stderr.write("Failed finding git workspace\n")
94         sys.exit(err)
96     return os.path.abspath(os.path.join(p.readlines()[0],
97                                         os.path.pardir))
100 def git_branch():
101     """Return the current git branch"""
103     p = git('branch')
105     if not p:
106         sys.stderr.write("Failed finding git branch\n")
107         sys.exit(err)
109     for elt in p:
110         if elt[0] == '*':
111             if elt.endswith('(no branch)'):
112                 return None
113             return elt.split()[1]
116 def git_parent_branch(branch):
117     """Return the parent of the current git branch.
119     If this branch tracks a remote branch, return the remote branch which is
120     tracked. If not, default to origin/master."""

```

```

122     if not branch:
123         return None

125     p = git("for-each-ref --format=%(refname:short) %(upstream:short) " +
126            "refs/heads/")

128     if not p:
129         sys.stderr.write("Failed finding git parent branch\n")
130         sys.exit(err)

132     for line in p:
133         # Git 1.7 will leave a ' ' trailing any non-tracking branch
134         if ' ' in line and not line.endswith('\n'):
135             local, remote = line.split()
136             if local == branch:
137                 return remote
138     return 'origin/master'

141 def git_comments(parent):
142     """Return a list of any checkin comments on this git branch"""

144     p = git('log --pretty=tformat:%%B:SEP: %s..' % parent)

146     if not p:
147         sys.stderr.write("Failed getting git comments\n")
148         sys.exit(err)

150     return [x.strip() for x in p.readlines() if x != ':SEP:\n']

153 def git_file_list(parent, paths=None):
154     """Return the set of files which have ever changed on this branch.

156     NB: This includes files which no longer exist, or no longer actually
157     differ."""

159     p = git("log --name-only --pretty=format: %s.. %s" %
160            (parent, ' '.join(paths)))

162     if not p:
163         sys.stderr.write("Failed building file-list from git\n")
164         sys.exit(err)

166     ret = set()
167     for fname in p:
168         if fname and not fname.isspace() and fname not in ret:
169             ret.add(fname.strip())

171     return ret

174 def not_check(root, cmd):
175     """Return a function which returns True if a file given as an argument
176     should be excluded from the check named by 'cmd'"""

178     ignorefiles = filter(os.path.exists,
179                          [os.path.join(root, ".git", "%s.NOT" % cmd),
180                           os.path.join(root, "exception_lists", cmd)])
181     if len(ignorefiles) > 0:
182         return ignore.ignore(root, ignorefiles, sys.stderr.write)
183     else:
184         return lambda x: False

187 def gen_files(root, parent, paths, exclude):

```

```

188     """Return a function producing file names, relative to the current
189     directory, of any file changed on this branch (limited to 'paths' if
190     requested), and excluding files for which exclude returns a true value """

192     # Taken entirely from Python 2.6's os.path.relpath which we would use if we
193     # could.
194     def relpath(path, here):
195         c = os.path.abspath(os.path.join(root, path)).split(os.path.sep)
196         s = os.path.abspath(here).split(os.path.sep)
197         l = len(os.path.commonprefix((s, c)))
198         return os.path.join(*[os.path.pardir] * (len(s)-l) + c[l:])

200     def ret(select=None):
201         if not select:
202             select = lambda x: True

204         for f in git_file_list(parent, paths):
205             f = relpath(f, '.')
206             if (os.path.exists(f) and select(f) and not exclude(f)):
207                 yield f
208     return ret

211 def comchk(root, parent, flist, output):
212     output.write("Comments:\n")

214     return Comments.comchk(git_comments(parent), check_db=True,
215                            output=output)

218 def mapfilechk(root, parent, flist, output):
219     ret = 0

221     # We are interested in examining any file that has the following
222     # in its final path segment:
223     # - Contains the word 'mapfile'
224     # - Begins with 'map.'
225     # - Ends with '.map'
226     # We don't want to match unless these things occur in final path segment
227     # because directory names with these strings don't indicate a mapfile.
228     # We also ignore files with suffixes that tell us that the files
229     # are not mapfiles.
230     MapfileRE = re.compile(r'.*((mapfile[^\/*]|(/map\.[^\/*]|(\.map)))$',
231                            re.IGNORECASE)
232     NotMapSuffixRE = re.compile(r'.*\.[ch]$', re.IGNORECASE)

234     output.write("Mapfile comments:\n")

236     for f in flist(lambda x: MapfileRE.match(x) and not
237                   NotMapSuffixRE.match(x)):
238         fh = open(f, 'r')
239         ret |= Mapfile.mapfilechk(fh, output=output)
240         fh.close()
241     return ret

244 def copyright(root, parent, flist, output):
245     ret = 0
246     output.write("Copyrights:\n")
247     for f in flist():
248         fh = open(f, 'r')
249         ret |= Copyright.copyright(fh, output=output)
250         fh.close()
251     return ret

```

```

254 def hdrchk(root, parent, flist, output):
255     ret = 0
256     output.write("Header format:\n")
257     for f in flist(lambda x: x.endswith('.h')):
258         fh = open(f, 'r')
259         ret |= HdrChk.hdrchk(fh, lenient=True, output=output)
260     fh.close()
261     return ret

264 def cstyle(root, parent, flist, output):
265     ret = 0
266     output.write("C style:\n")
267     for f in flist(lambda x: x.endswith('.c') or x.endswith('.h')):
268         fh = open(f, 'r')
269         ret |= CStyle.cstyle(fh, output=output, picky=True,
270                             check_posix_types=True,
271                             check_continuation=True)
272     fh.close()
273     return ret

276 def jstyle(root, parent, flist, output):
277     ret = 0
278     output.write("Java style:\n")
279     for f in flist(lambda x: x.endswith('.java')):
280         fh = open(f, 'r')
281         ret |= JStyle.jstyle(fh, output=output, picky=True)
282     fh.close()
283     return ret

286 def manlint(root, parent, flist, output):
287     ret = 0
288     output.write("Man page format:\n")
289     ManfileRE = re.compile(r'.*[0-9][a-z]*$', re.IGNORECASE)
290     for f in flist(lambda x: ManfileRE.match(x)):
291         fh = open(f, 'r')
292         ret |= ManLint.manlint(fh, output=output, picky=True)
293     fh.close()
294     return ret

296 def keywords(root, parent, flist, output):
297     ret = 0
298     output.write("SCCS Keywords:\n")
299     for f in flist():
300         fh = open(f, 'r')
301         ret |= Keywords.keywords(fh, output=output)
302     fh.close()
303     return ret

306 def run_checks(root, parent, cmds, paths='', opts={}):
307     """Run the checks given in 'cmds', expected to have well-known signatures,
308     and report results for any which fail.

310     Return failure if any of them did.

312     NB: the function name of the commands passed in is used to name the NOT
313     file which excepts files from them."""

315     ret = 0

317     for cmd in cmds:
318         s = StringIO()

```

```

320         exclude = not_check(root, cmd.func_name)
321         result = cmd(root, parent, gen_files(root, parent, paths, exclude),
322                    output=s)
323         ret |= result

325         if result != 0:
326             print s.getvalue()

328     return ret

331 def nits(root, parent, paths):
332     cmds = [copyright,
333            cstyle,
334            hdrchk,
335            jstyle,
336            keywords,
337            manlint,
338            mapfilechk]
339     run_checks(root, parent, cmds, paths)

342 def pbchk(root, parent, paths):
343     cmds = [comchk,
344            copyright,
345            cstyle,
346            hdrchk,
347            jstyle,
348            keywords,
349            manlint,
350            mapfilechk]
351     run_checks(root, parent, cmds)

354 def main(cmd, args):
355     parent_branch = None

357     try:
358         opts, args = getopt.getopt(args, 'b:')
359     except getopt.GetoptError, e:
360         sys.stderr.write(str(e) + '\n')
361         sys.stderr.write("Usage: %s [-b branch] [path...]\n" % cmd)
362         sys.exit(1)

364     for opt, arg in opts:
365         if opt == '-b':
366             parent_branch = arg

368     if not parent_branch:
369         parent_branch = git_parent_branch(git_branch())

371     func = nits
372     if cmd == 'git-pbchk':
373         func = pbchk
374     if args:
375         sys.stderr.write("only complete workspaces may be pbchk'd\n");
376         sys.exit(1)

378     func(git_root(), parent_branch, args)

380 if __name__ == '__main__':
381     try:
382         main(os.path.basename(sys.argv[0]), sys.argv[1:])
383     except GitError, e:
384         sys.stderr.write("failed to run git:\n%s\n" % str(e))
385         sys.exit(1)

```