

```
*****
11225 Fri Feb 2 23:18:32 2018
new/usr/src/pkg/manifests/developer-build-onbld.mf
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
*****
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 #

22 #
23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright 2015, Richard Lowe
25 # Copyright 2012, Piotr Jasiukajtis
26 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
27 # Copyright (c) 2014, Joyent, Inc.
28 # Copyright 2016 Toomas Soome <tsoome@me.com>
29 #

31 set name=pkg.fmri value=pkg:/developer/build/onbld@$(PKGVERS)
32 set name=pkg.description value="tools used to build the OS-Net consolidation"
33 set name=pkg.summary value="OS-Net Build Tools"
34 set name=info.classification \
35     value="org.opensolaris.category.2008:Development/Distribution Tools"

37 #
38 # This package should not be incorporated. This allows the tools
39 # to be upgraded without upgrading the entire system.
40 #
41 set name=org.opensolaris.noincorp value=true
42 set name=variant.arch value=$(ARCH)
43 dir path=opt group=sys
44 dir path=opt/onbld
45 dir path=opt/onbld/bin
46 dir path=opt/onbld/bin/$(ARCH)
47 dir path=opt/onbld/env
48 dir path=opt/onbld/etc
49 dir path=opt/onbld/etc/exception_lists
50 dir path=opt/onbld/lib
51 dir path=opt/onbld/lib/$(ARCH)
52 dir path=opt/onbld/lib/$(ARCH)/64
53 dir path=opt/onbld/lib/perl
54 dir path=opt/onbld/lib/python$(PYTHON_VERSION)
55 dir path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld
56 dir path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks
57 dir path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm
58 dir path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/hgext
```

```
58 dir path=opt/onbld/man
59 dir path=opt/onbld/man/manlsonbld
60 dir path=opt/onbld/share
61 $(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/aw mode=0555
62 $(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/btxld mode=0555
63 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/chk4ubin mode=0555
64 file path=opt/onbld/bin/$(ARCH)/codereview mode=0555
65 file path=opt/onbld/bin/$(ARCH)/cscope-fast mode=0555
66 file path=opt/onbld/bin/$(ARCH)/ctfconvert mode=0555
67 file path=opt/onbld/bin/$(ARCH)/ctfdump mode=0555
68 file path=opt/onbld/bin/$(ARCH)/ctfmerge mode=0555
69 file path=opt/onbld/bin/$(ARCH)/ctfstabs mode=0555
70 file path=opt/onbld/bin/$(ARCH)/ctfstrip mode=0555
71 file path=opt/onbld/bin/$(ARCH)/cv mode=0555
72 $(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/elfextract mode=0555
73 file path=opt/onbld/bin/$(ARCH)/findref mode=0555
74 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/forth mode=0555
75 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/forth_preload.so.1 mode=0555
76 file path=opt/onbld/bin/$(ARCH)/install mode=0555
77 file path=opt/onbld/bin/$(ARCH)/lintdump mode=0555
78 file path=opt/onbld/bin/$(ARCH)/localizedef mode=0555
79 file path=opt/onbld/bin/$(ARCH)/make mode=0555
80 $(i386_ONLY)file path=opt/onbld/bin/$(ARCH)/mbh_patch mode=0555
81 file path=opt/onbld/bin/$(ARCH)/ndrgen mode=0555
82 file path=opt/onbld/bin/$(ARCH)/ndrgen mode=0555
83 file path=opt/onbld/bin/$(ARCH)/protocmp mode=0555
84 file path=opt/onbld/bin/$(ARCH)/protolist mode=0555
85 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/stabs mode=0555
86 file path=opt/onbld/bin/$(ARCH)/tic mode=0555
87 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/tokenize mode=0555
88 $(sparc_ONLY)file path=opt/onbld/bin/$(ARCH)/tokenize.exe mode=0555
89 file path=opt/onbld/bin/$(ARCH)/zic mode=0555
90 file path=opt/onbld/bin/Install mode=0555
91 file path=opt/onbld/bin/bldenv mode=0555
92 file path=opt/onbld/bin/bringovercheck mode=0555
93 file path=opt/onbld/bin/build_cscope mode=0555
94 file path=opt/onbld/bin/cddlchk mode=0555
95 file path=opt/onbld/bin/check_rtime mode=0555
96 file path=opt/onbld/bin/checkpaths mode=0555
97 file path=opt/onbld/bin/copyrightchk mode=0555
98 file path=opt/onbld/bin/cstyle mode=0555
99 file path=opt/onbld/bin/elfcmp mode=0555
100 file path=opt/onbld/bin/find_elf mode=0555
101 file path=opt/onbld/bin/findcrypto mode=0555
102 file path=opt/onbld/bin/flg.flp mode=0555
103 file path=opt/onbld/bin/genoffsets mode=0555
104 file path=opt/onbld/bin/git-pbchk mode=0555
105 file path=opt/onbld/bin/hdrchk mode=0555
107 file path=opt/onbld/bin/hg-active mode=0555
108 file path=opt/onbld/bin/hgsetup mode=0555
106 file path=opt/onbld/bin/interface_check mode=0555
107 file path=opt/onbld/bin/interface_cmp mode=0555
108 file path=opt/onbld/bin/jstyle mode=0555
109 file path=opt/onbld/bin/mapfilechk mode=0555
110 file path=opt/onbld/bin/nightly mode=0555
111 file path=opt/onbld/bin/onu mode=0555
112 file path=opt/onbld/bin/protocmp.terse mode=0555
113 file path=opt/onbld/bin/sccscheck mode=0555
114 file path=opt/onbld/bin/signit mode=0555
115 file path=opt/onbld/bin/signproto mode=0555
116 file path=opt/onbld/bin/validate_flg mode=0555
117 file path=opt/onbld/bin/validate_paths mode=0555
118 file path=opt/onbld/bin/validate_pkg mode=0555
119 file path=opt/onbld/bin/wdiff mode=0555
120 file path=opt/onbld/bin/webrev mode=0555
121 file path=opt/onbld/bin/which_scm mode=0555
```

```

122 file path=opt/onbld/bin/ws mode=0555
123 file path=opt/onbld/bin/wsdiff mode=0555
124 file path=opt/onbld/bin/xref mode=0555
125 file path=opt/onbld/bin/xref.mk
126 file path=opt/onbld/env/illumos
127 file path=opt/onbld/etc/exception_lists/check_rtime
128 file path=opt/onbld/etc/exception_lists/interface_check
129 file path=opt/onbld/etc/exception_lists/interface_cmp
130 file path=opt/onbld/etc/its.conf
131 file path=opt/onbld/etc/its.reg
132 file path=opt/onbld/lib/$(ARCH)/64/libmakestate.so.1
133 file path=opt/onbld/lib/$(ARCH)/libdwarf.so.1
134 file path=opt/onbld/lib/$(ARCH)/libmakestate.so.1
135 file path=opt/onbld/lib/perl/onbld_elfmod.pm
136 file path=opt/onbld/lib/perl/onbld_elfmod_vertype.pm
137 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/CStyle.py \
138     mode=0444
139 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/CStyle.pyc \
140     mode=0444
141 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Cddl.py mode=0444
142 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Cddl.pyc \
143     mode=0444
144 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/CmtBlk.py \
145     mode=0444
146 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/CmtBlk.pyc \
147     mode=0444
148 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Comments.py \
149     mode=0444
150 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Comments.pyc \
151     mode=0444
152 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Copyright.py \
153     mode=0444
154 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Copyright.pyc \
155     mode=0444
156 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/DbLookups.py \
157     mode=0444
158 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/DbLookups.pyc \
159     mode=0444
160 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/HdrChk.py \
161     mode=0444
162 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/HdrChk.pyc \
163     mode=0444
164 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/JStyle.py \
165     mode=0444
166 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/JStyle.pyc \
167     mode=0444
168 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Keywords.py \
169     mode=0444
170 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Keywords.pyc \
171     mode=0444
172 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/ManLint.py \
173     mode=0444
174 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/ManLint.pyc \
175     mode=0444
176 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Mapfile.py \
177     mode=0444
178 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/Mapfile.pyc \
179     mode=0444
180 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/ProcessCheck.py \
181     mode=0444
182 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/ProcessCheck.pyc \
183     mode=0444
184 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/SpellCheck.py \
185     mode=0444
186 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/SpellCheck.pyc \

```

```

187     mode=0444
188 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/_init__.py \
189     mode=0444
190 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Checks/_init__.pyc \
191     mode=0444
192 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/Backup.py mode=0444
193 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/Ignore.py mode=0444
194 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/Ignore.pyc mode=0444
195 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/Version.py mode=0444
196 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/Version.pyc \
197     mode=0444
198 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/_init__.py mode=0444
199 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/_init__.pyc mode=0444
200 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/hgext/_init__.py \
201     mode=0444
202     mode=0444
203 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/WorkSpace.py \
204     mode=0444
205 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/WorkSpace.pyc \
206     mode=0444
207 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/Scm/_init__.py \
208     mode=0444
209 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/_init__.pyc \
210     mode=0444
211 file path=opt/onbld/man/man1onbld/Install.1onbld
212 file path=opt/onbld/man/man1onbld/bldenv.1onbld
213 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/hgext/_init__.py \
214     mode=0444
215 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/hgext/_init__.pyc \
216     mode=0444
217 file path=opt/onbld/lib/python$(PYTHON_VERSION)/onbld/hgext/cdm.py mode=0444
218 file path=opt/onbld/man/man1onbld/_.1onbld
219 file path=opt/onbld/man/man1onbld/_.1onbld
220 file path=opt/onbld/man/man1onbld/bringovercheck.1onbld
221 file path=opt/onbld/man/man1onbld/btx1d.1onbld
222 file path=opt/onbld/man/man1onbld/cddlchk.1onbld
223 file path=opt/onbld/man/man1onbld/check_rtime.1onbld
224 file path=opt/onbld/man/man1onbld/checkpaths.1onbld
225 file path=opt/onbld/man/man1onbld/codereview.1onbld
226 file path=opt/onbld/man/man1onbld/cstyle.1onbld
227 file path=opt/onbld/man/man1onbld/cw.1onbld
228 file path=opt/onbld/man/man1onbld/find_elf.1onbld
229 file path=opt/onbld/man/man1onbld/findunref.1onbld
230 file path=opt/onbld/man/man1onbld/flg.flp.1onbld
231 file path=opt/onbld/man/man1onbld/git-pbchk.1onbld
232 file path=opt/onbld/man/man1onbld/hdrchk.1onbld
233 file path=opt/onbld/man/man1onbld/hgsetup.1onbld
234 file path=opt/onbld/man/man1onbld/interface_check.1onbld
235 file path=opt/onbld/man/man1onbld/interface_cmp.1onbld
236 file path=opt/onbld/man/man1onbld/jstyle.1onbld
237 file path=opt/onbld/man/man1onbld/lintdump.1onbld
238 file path=opt/onbld/man/man1onbld/mapfilechk.1onbld
239 file path=opt/onbld/man/man1onbld/nrigen.1onbld
240 file path=opt/onbld/man/man1onbld/nightly.1onbld
241 file path=opt/onbld/man/man1onbld/onu.1onbld
242 file path=opt/onbld/man/man1onbld/scscheck.1onbld
243 file path=opt/onbld/man/man1onbld/signit.1onbld
244 file path=opt/onbld/man/man1onbld/signproto.1onbld
245 file path=opt/onbld/man/man1onbld/webrev.1onbld
246 file path=opt/onbld/man/man1onbld/which_scm.1onbld
247 file path=opt/onbld/man/man1onbld/ws.1onbld
248 file path=opt/onbld/man/man1onbld/wsdiff.1onbld
249 file path=opt/onbld/man/man1onbld/xref.1onbld
250 file path=opt/onbld/share/make.rules mode=0444
251 hardlink path=opt/onbld/bin/$(ARCH)/install.bin target=../install
252 legacy pkg=SUNWnbld desc="tools used to build the OS-Net consolidation" \
253     name="OS-Net Build Tools" version=11.11,REV=2009.10.22
254 license cr_Sun license=cr_Sun
255 license lic_CDDL license=lic_CDDL
256 license usr/src/tools/ctf/dwarf/THIRDPARTYLICENSE \

```

```
238     license=usr/src/tools/ctf/dwarf/THIRDPARTYLICENSE
239 license usr/src/tools/onbld/THIRDPARTYLICENSE \
240     license=usr/src/tools/onbld/THIRDPARTYLICENSE
241 link path=opt/onbld/bin/${ARCH}/dmake target=make
242 link path=opt/onbld/bin/git-nits target=git-pbchk
243 link path=opt/onbld/lib/python target=python${PYTHON_VERSION}
244 link path=opt/onbld/man/man1onbld/git-nits.lonbld target=git-pbchk.lonbld
245 # webrev(1) requires ps2pdf
246 depend fmri=print/filter/ghostscript type=require
247 # nightly(1) uses wget
248 depend fmri=web/wget type=require
```

new/usr/src/tools/Makefile

```
*****
3201 Fri Feb 2 23:18:33 2018
new/usr/src/tools/Makefile
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
*****
```

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 #

22 #
23 # Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
25 # Copyright 2016 Toomas Soome <tsoome@me.com>
26 # Copyright (c) 2016, Chris Fraire <cfraire@me.com>.
27 #

29 include ../Makefile.master

31 # Bootstrap problem --
32 # 'cw' must be built before anything else can be built.

34 BOOT_SUBDIRS= \
35 cw

37 COMMON_SUBDIRS= \
38 codereview \
39 codesign \
40 cscope-fast \
41 ctf \
42 env \
43 findunref \
44 install.bin \
45 lintdump \
46 make \
47 ndrgen \
48 onbld \
49 protocmp \
50 protolist \
51 scripts

53 #
54 # special versions of commands for use only in build
55 #
56 UNSHIPPED_SUBDIRS = \
57 localedef \
58 mandoc \
59 tic \
60 vtfontcvt \
61 zic
63 sparc_SUBDIRS= \
64 chk4ubin \
65 stabs \
66 tokenize
68 i386_SUBDIRS= \
69 aw \
70 elfextract \
71 mbh_patch \
72 btxld
74 LINTSUBDIRS= \
75 codereview \
76 ctf \
77 cw \
78 findunref \
79 lintdump \
80 ndrgen \
81 protocmp \
82 protolist
84 SUBDIRS= \
85 \$(MACH)_SUBDIRS \
86 \$(COMMON_SUBDIRS) \
87 \$(UNSHIPPED_SUBDIRS)
89 include Makefile.tools
91 ROOTDIRS= \
92 \$(ROOTOPT) \
93 \$(ROOTONBLD) \
94 \$(ROOTONBLD)/bin \
95 \$(ROOTONBLD)/bin/\$(MACH) \
96 \$(ROOTONBLD)/lib \
97 \$(ROOTONBLD)/lib/\$(MACH) \
98 \$(ROOTONBLD)/lib/\$(MACH)/64 \
99 \$(ROOTONBLD)/lib/perl \
100 \$(ROOTONBLD)/lib/python\$(PYTHON_VERSION) \
101 \$(ROOTONBLD)/lib/python\$(PYTHON_VERSION)/onbld \
102 \$(ROOTONBLD)/lib/python\$(PYTHON_VERSION)/onbld/Checks \
103 \$(ROOTONBLD)/lib/python\$(PYTHON_VERSION)/onbld/hgext \
104 \$(ROOTONBLD)/lib/python\$(PYTHON_VERSION)/onbld/Scm \
105 \$(ROOTONBLD)/env \
106 \$(ROOTONBLD)/etc \
107 \$(ROOTONBLD)/etc/exception_lists \
108 \$(ROOTONBLD)/share \
109 \$(ROOTONBLD)/man \
110 \$(ROOTONBLD)/man/manlonbld
111 all := TARGET= install
112 install := TARGET= install
113 clean := TARGET= clean
114 clobber := TARGET= clobber
115 lint := TARGET= lint
116 _msg := TARGET= _msg
118 .KEEP_STATE:
120 #
121 # Only create directories in the tools proto area when doing an actual
122 # build, not a clean or clobber.
123 #

1

new/usr/src/tools/Makefile

```
*****
59     tic \  
60     vtfontcvt \  
61     zic  
63 sparc_SUBDIRS= \  
64     chk4ubin \  
65     stabs \  
66     tokenize  
68 i386_SUBDIRS= \  
69     aw \  
70     elfextract \  
71     mbh_patch \  
72     btxld  
74 LINTSUBDIRS= \  
75     codereview \  
76     ctf \  
77     cw \  
78     findunref \  
79     lintdump \  
80     ndrgen \  
81     protocmp \  
82     protolist  
84 SUBDIRS= \  
85     $(MACH)_SUBDIRS \  
86     $(COMMON_SUBDIRS) \  
87     $(UNSHIPPED_SUBDIRS)  
89 include Makefile.tools  
91 ROOTDIRS= \  
92     $(ROOTOPT) \  
93     $(ROOTONBLD) \  
94     $(ROOTONBLD)/bin \  
95     $(ROOTONBLD)/bin/$(MACH) \  
96     $(ROOTONBLD)/lib \  
97     $(ROOTONBLD)/lib/$(MACH) \  
98     $(ROOTONBLD)/lib/$(MACH)/64 \  
99     $(ROOTONBLD)/lib/perl \  
100    $(ROOTONBLD)/lib/python$(PYTHON_VERSION) \  
101    $(ROOTONBLD)/lib/python$(PYTHON_VERSION)/onbld \  
102    $(ROOTONBLD)/lib/python$(PYTHON_VERSION)/onbld/Checks \  
103    $(ROOTONBLD)/lib/python$(PYTHON_VERSION)/onbld/hgext \  
104    $(ROOTONBLD)/lib/python$(PYTHON_VERSION)/onbld/Scm \  
105    $(ROOTONBLD)/env \  
106    $(ROOTONBLD)/etc \  
107    $(ROOTONBLD)/etc/exception_lists \  
108    $(ROOTONBLD)/share \  
109    $(ROOTONBLD)/man \  
110    $(ROOTONBLD)/man/manlonbld  
111 all := TARGET= install  
112 install := TARGET= install  
113 clean := TARGET= clean  
114 clobber := TARGET= clobber  
115 lint := TARGET= lint  
116 _msg := TARGET= _msg  
118 .KEEP_STATE:  
120 #  
121 # Only create directories in the tools proto area when doing an actual  
122 # build, not a clean or clobber.  
123 #
```

2

```
124 DOROOTDIRS= $(ROOTDIRS)
125 clobber:= DOROOTDIRS=
126 clean:= DOROOTDIRS=

128 DOROOTONBLDLIBPY= $(ROOTONBLDLIBPY)
129 clobber:= DOROOTONBLDLIBPY=
130 clean:= DOROOTONBLDLIBPY=

132 all install: $(SUBDIRS)

134 clean: $(SUBDIRS)

136 clobber: $(SUBDIRS)
137     $(RM) -rf $(TOOLS_PROTO)

139 lint: $(LINTSUBDIRS)

141 _msg: $(MSGSUBDIRS)

143 .PARALLEL: $(SUBDIRS) $(CLOSED_SUBDIRS)

145 $(SUBDIRS) $(CLOSED_SUBDIRS): $(BOOT_SUBDIRS)

147 $(BOOT_SUBDIRS) $(SUBDIRS): $$($(DOROOTDIRS) $$($(DOROOTONBLDLIBPY)) FRC
148     @cd $@; pwd; $(MAKE) $(TARGET)

150 $(ROOTDIRS):
151     $(INS.dir)

153 $(ROOTONBLDLIBPY): $(ROOTDIRS)
154     $(RM) -r $@; $(SYMLINK) python$(PYTHON_VERSION) $@

156 make: ctf

158 FRC:
```

new/usr/src/tools/README.tools

9944 Fri Feb 2 23:18:33 2018

new/usr/src/tools/README.tools

9001 cdm is useless, remove it

9002 webrev should know how to get the git user name

Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>

Reviewed by: Joshua M. Clulow <jmc@joyent.com>

```
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) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
```

```
25 This directory contains the tools used to do a full build of the
26 OS/Net workspace. They usually live in the /opt/onbld directory on build
27 machines. From here, 'make install' will build and install the tools
28 in $ROOT/opt/onbld.
```

30 Layout of /opt/onbld

31 -----

```
33 /opt/onbld/etc/abi
34     contains Solaris ABI database (ABI_*.db) and exceptions
35     for ABI Auditing tool (interface_check, interface_cmp).
```

```
37 /opt/onbld/bin
38     basic bin directory - contains scripts.
```

```
40 /opt/onbld/bin/${MACH}
41     architecture-specific bin directory for binaries.
```

```
43 /opt/onbld/env
44     build environment files.
```

```
46 /opt/onbld/lib
47     libraries used by the build tools.
```

```
49 /opt/onbld/lib/python<version>/
50     python modules used by the build tools.
```

```
52 /opt/onbld/lib/python<version>/onbld/hgext
53     Mercurial extensions.
```

```
52 /opt/onbld/lib/python/
53     symlink to the modules directory of the currently preferred
54     python version.
55     python version. This exists to retain compatibility both for
```

1

new/usr/src/tools/README.tools

```
56 /opt/onbld/man
57     rudimentary man pages for some of the tools.
```

```
60 Tool Summary
61 -----
```

```
63 bldenv
64     companion to 'nightly.' Takes the same environment file you
65     used with 'nightly,' and starts a shell with the environment
66     set up the same way as 'nightly' set it up. This is useful
67     if you're trying to quickly rebuild portions of a workspace
68     built by 'nightly'. 'ws' should not be used for this since it
69     sets the environment up differently and may cause everything
70     to rebuild (because of different -I or -L paths).
```

```
72 build_cscope
73     builds cscope databases in the uts, the platform subdirectories
74     of uts, and in usr/src. Uses cscope-fast.
```

```
82 cdm
83     A Mercurial extension providing various commands useful for ON
84     development
```

```
76 check_rtime
77     checks ELF attributes used by ELF dynamic objects in the proto area.
78     Used by 'nightly's -r option, to check a number of ELF runtime
79     attributes for consistency with common build rules. nightly uses
80     the -o option to simplify the output for diffing with previous
81     build results. It also uses the -i option to obtain NEEDED and RUNPATH
82     entries, which help detect changes in software dependencies and makes
83     sure objects don't have any strange runpaths like /opt/SUNWspro/lib.
```

```
85 codereview
86     Given two filenames, creates a postscript file with the file
87     differences highlighted.
```

```
89 codesign
90     Tools for signing cryptographic modules using the official
91     Sun release keys stored on a remote signing server. This
92     directory contains signit, a client program for signing
93     files with the signing server; signproto, a shell script
94     that finds crypto modules in $ROOT and signs them using
95     signit; and codesign_server.pl, the code that runs on the
96     server. The codesign_server code is not used on an ON
97     build machine but is kept here for source control purposes.
```

```
99 copyrightchk
100    Checks that files have appropriate SMI copyright notices.
101    Primarily used by wx
```

```
103 cscope-fast
104    The fast version of cscope that we use internally. Seems to work,
105    but may need more testing before it's placed in the gate. The source
106    just really needs to be here.
```

```
107 108 cstyle
109    checks C source for compliance with OS/Net guidelines.
```

```
111 ctfconvert
112    Convert symbolic debugging information in an object file to the Compact
113    ANSI-C Type Format (CTF).
```

2

```

115 ctfdump
116     Decode and display CTF data stored in a raw file or in an ELF file.

118 ctfmerge
119     Merge the CTF data from one or more object files.

121 elfcmp
122     Compares two ELF modules (e.g. .o files, executables) section by
123     section. Useful for determining whether "trivial" changes -
124     cstyle, lint, etc - actually changed the code. The -S option
125     is used to test whether two binaries are the same except for
126     the elfsign signature.

128 find_elf
129     Search a directory tree for ELF objects, and produce one line of
130     output per object. Used by check_rtime and interface_check to locate
131     the objects to examine.

133 findunref
134     Finds all files in a source tree that have access times older than a
135     certain time and are not in a specified list of exceptions. Since
136     'nightly' timestamps the start of the build, and findunref uses its
137     timestamp (by default), this can be used to find all files that were
138     unreferenced during a nightly build). Since some files are only used
139     during a SPARC or Intel build, 'findunref' needs to be run on
140     workspaces from both architectures and the results need to be merged.
141     For instance, if $INTELSRC and $SPARCSRC are set to the usr/src
142     directories of your Intel and SPARC nightly workspaces, then you
143     can merge the results like so:

145     $ findunref $INTELSRC $INTELSRC/tools/findunref/exception_list | \
146         sort > ~/unref-i386.out
147     $ findunref $SPARCSRC $SPARCSRC/tools/findunref/exception_list | \
148         sort > ~/unref-sparc.out
149     $ comm -12 ~/unref-i386.out ~/unref-sparc.out > ~/unref.out

151 hdrchk
152     checks headers for compliance with OS/Net standards (form, includes,
153     C++ guards).

165 hgsetup
166     creates a basic Mercurial configuration for the user.

168 hg-active
169     helper used by webrev to generate file lists for Mercurial
170     workspaces.

155 install.bin
156     binary version of /usr/sbin/install. Used to be vastly faster
157     (since /usr/sbin/install is a shell script), but may only be a bit
158     faster now. One speedup includes avoiding the name service for the
159     well-known, never-changing password entries like 'root' and 'sys.'

161 interface_check
162     detects and reports invalid versioning in ELF objects.
163     Optionally generates an interface description file for
164     the workspace.

166 interface_cmp
167     Compares two interface description files, as produced by
168     interface_check, and flags invalid deviations in ELF object
169     versioning between them. interface_cmp can be used between Solaris
170     gates to ensure that older releases remain compatible with the
171     development gate. It can also be used to validate new changes to
172     the development gate before they are integrated.

```

```

174 lntdump
175     dumps the contents of one or more lint libraries; see lntdump(1)

177 ndrgen
178     Network Data Language (NDL) RPC protocol compiler to support DCE
179     RPC/MSRPC and SMB/CIFS. ndrgen takes an input protocol definition
180     file (say, proto.ndl) and generates an output C source file
181     (proto_ndr.c) containing the Network Data Representation (NDR)
182     marshalling routines to implement the RPC protocol.

184 nightly
185     nightly build script. Takes an environment (or 'env') file describing
186     such things as the workspace, the parent, and what to build. See
187     env/developer and env/gatekeeper for sample, hopefully well-commented
188     env files.

190 protocmp
191     compares proto lists and the package definitions. Used by nightly
192     to determine if the proto area matches the packages, and to detect
193     differences between a child's proto area and a parents.

195 protocmp.terse
196     transforms the output of protocmp into something a bit more friendly

198 protolist
199     create a list of what's in the proto area, to feed to protocmp.

202 ws
203     creates a shell with the environment set up to build in the given
204     workspace. Used mostly for non-full-build workspaces, so it sets up
205     to pull headers and libraries from the proto area of the parent if
206     they aren't in the child's proto area.

208 tokenize
209     Used to build the sun4u boot block.

211 webrev
212     Generates a set of HTML pages that show side-by-side diffs of
213     changes in your workspace, for easy communication of code
214     review materials. Can automatically find edited files or use a
215     manually-generated list; knows how to use wx's active file for
216     lists of checked-out files and proposed SCCS comments.

218 which_scm
219     Reports the current Source Code Management (SCM) system in use
220     and the top-level directory of the workspace.

222 wsdiff
223     Detect object differences between two ON proto areas. Used by
224     nightly(1) to determine what changed between two builds. Handy
225     for identifying the set of built objects impacted by a given
226     source change. This information is needed for patch construction.

229 How to do a full build
230 -----
232 1. Find an environment file that might do what you want to do. If you're just
233     a developer wanting to do a full build in a child of the gate, copy the
234     'developer' environment file to a new name (private to you and/or the
235     work being done in this workspace, to avoid collisions with others). Then
236     edit the file and tailor it to your workspace. Remember that this file
237     is a shell script, so it can do more than set environment variables.

```

```
239 2. Run 'nightly' and give it your environment file as an
240   option. 'nightly' will first look for your environment file in
241   /opt/onbld/env, and if it's not there then it will look for it as an
242   absolute or relative path. Some people put their environment files in
243   their workspace to keep them close.
245 3. When 'nightly' is complete, it will send a summary of what happened to
246   $MAILTO. Usually, the less info in the mail the better. If you have failures,
247   you can go look at the full log of what happened, generally in
248   $CODEMGR_WS/log/log.<date>/nightly.log (the mail_msg it sent and the proto
249   list are there too). You can also find the individual build logs, like
250   'make clobber' and 'make install' output in $SRC, under names like
251   clobber-${MACH}.out and install-${MACH}.out (for a DEBUG build). These
252   will be smaller than nightly.log, and maybe more searchable.
```

```
254 Files you have to update to add a tool
255 -----
256
```

```
257 1. Add the tool in its appropriate place.
258 2. Update the Makefile as required.
259 3. Update usr/src/pkg/manifests/developer-build-onbld.mf
260 4. Update usr/src/tools/README.tools (this file).
261 5. Repeat 1-4 for any man pages.
```

new/usr/src/tools/onbld/Makefile

1394 Fri Feb 2 23:18:34 2018

new/usr/src/tools/onbld/Makefile

9001 cdm is useless, remove it

9002 webrev should know how to get the git user name

Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>

Reviewed by: Joshua M. Clulow <jmc@joyent.com>

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 #

22 #
23 # Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
24 #

26 include \$(SRC)/Makefile.master
27 include ../Makefile.tools

29 SUBDIRS= \
30 Checks \
31 hgext \
31 Scm

33 PYSRCS = \
34 __init__.py

36 PYOJJS = \$(PYSRCS:.py=% .pyc)
37 PYTOPDIR = \$(ROOTONBLDLIB)
38 PYMODDIR = onbld

40 all := TARGET = all
41 install := TARGET = install
42 clean := TARGET = clean
43 clobber := TARGET = clobber

45 include ../Makefile.python

47 .KEEP_STATE:

49 all: \$(PYVERSOJJS) \$(SUBDIRS)

51 install: all \$(ROOTPYFILES) \$(SUBDIRS)

53 clean: \$(SUBDIRS)

55 clobber: clean pyclobber \$(SUBDIRS)

57 \$(SUBDIRS): FRC

1

new/usr/src/tools/onbld/Makefile

58 @cd \$@; pwd; \$(MAKE) \$(TARGET)

60 FRC:

2

```
new/usr/src/tools/onbld/Scm/Makefile
```

```
1
```

```
*****
```

```
1210 Fri Feb 2 23:18:35 2018
```

```
new/usr/src/tools/onbld/Scm/Makefile
```

```
9001 cdm is useless, remove it
```

```
9002 webrev should know how to get the git user name
```

```
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
```

```
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
```

```
*****
```

```
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 #
```

```
22 #
23 # Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright (c) 2014, Joyent, Inc.
25 #
```

```
27 include $(SRC)/Makefile.master
28 include ../../Makefile.tools
```

```
30 PYSRCS = \
31     __init__.py \
32     Ignore.py \
33     Backup.py \
34     Ignore.py \
35     Version.py \
36     WorkSpace.py
```

```
34 PYOJJS =      $(PYSRCS:%.py=%.pyc)
35 PYTOPDIR =    $(ROOTONBLLIB)
36 PYMODDIR =    onbld/Scm
```

```
38 include ../../Makefile.python
```

```
40 all: $(PYVERSOJJS)
```

```
42 install: all $(ROOTPYFILES)
```

```
44 clean:
```

```
46 clobber: clean pyclobber
```

new/usr/src/tools/scripts/Makefile

```
*****
3904 Fri Feb 2 23:18:38 2018
new/usr/src/tools/scripts/Makefile
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
*****
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) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
23 #
24 # Copyright 2010, Richard Lowe
26 SHELL=/usr/bin/ksh93
28 SHFILES= \
29     Install \
30     bldenv \
31     build_cscope \
32     bringovercheck \
33     checkpaths \
34     cstyle \
35     elfcmp \
36     flg.flp \
37     genoffsets \
38     hgsetup \
39     nightly \
40     onu \
41     protocomp.terse \
42     sccscheck \
43     webrev \
44     which_scm \
45     ws \
46     xref
47 PERLFILES= \
48     check_rtime \
49     find_elf \
50     interface_check \
51     interface_cmp \
52     jstyle \
53     validate_flg \
54     validate_paths \
55     wdiff
57 PERLMODULES= \
```

1

new/usr/src/tools/scripts/Makefile

```
58     onbld_elfmod.pm \
59     onbld_elfmod_vertype.pm
62 PYFILES= \
63     cddlchk \
64     copyrightchk \
65     git-pbchk \
66     hdrchk \
67     hg-active \
68     mapfilechk \
69     validate_pkg \
70     wsdiff
71 SCRIPTLINKS= \
72     git-nits
74 MAN1ONBLDFILES= \
75     Install.lonbld \
76     bldenv.lonbld \
77     bringovercheck.lonbld \
78     cddlchk.lonbld \
79     checkpaths.lonbld \
80     check_rtime.lonbld \
81     cstyle.lonbld \
82     find_elf.lonbld \
83     flg.flp.lonbld \
84     git-pbchk.lonbld \
85     hdrchk.lonbld \
86     interface_check.lonbld \
87     interface_cmp.lonbld \
88     hgsetup.lonbld \
89     jstyle.lonbld \
90     mapfilechk.lonbld \
91     nightly.lonbld \
92     onu.lonbld \
93     sccscheck.lonbld \
94     webrev.lonbld \
95     which_scm.lonbld \
96     ws.lonbld \
97     wsdiff.lonbld \
98     xref.lonbld
99 MAN1ONBLDLINKS= \
100     git-nits.lonbld
102 MAKEFILES= \
103     xref.mk
105 ETCFILES= \
106     hgstyle \
107     its.conf \
108     its.reg
109 EXCEPTFILES= \
110     check_rtime \
111     interface_check \
112     interface_cmp
114 CLEANFILES = $(SHFILES) $(PERLFILES) bldenv.lonbld onu.sh
115 onu.sh: onu.sh.in
116         $(SED) -e "s:@PYTHON_VERSION@:$PYTHON_VERSION:g" < onu.sh.in > @@
117
118 include ..../Makefile.tools
119 include ..../Makefile.tools
```

2

```
121 ROOTONBLDSRIPTLINKS = $(SCRIPTLINKS:%=${ROOTONBLDBIN}/%)
122 ROOTONBLDMAN1ONBLDLINKS = $(MAN1ONBLDLINKS:%=${ROOTONBLDMAN1ONBLD}/%)

124 $(ROOTONBLDETCFILES)      := FILEMODE= 644
125 $(ROOTONBLDEXCEPTFILES)   := FILEMODE= 644
126 $(ROOTONBLDPERLMODULES)   := FILEMODE= 644
127 $(ROOTONBLDMAKEFILES)    := FILEMODE= 644
128 $(ROOTONBLDMAN1ONBLDFILES) := FILEMODE= 644

130 .KEEP_STATE:

132 all:    $(SHFILES) $(PERLFILES) $(PERLMODULES) $(PYFILES) \
133           $(MAN1ONBLDFILES) $(MAKEFILES)

135 $(ROOTONBLDBIN)/git-nits:
136     $(RM) $(ROOTONBLDBIN)/git-nits
137     $(SYMLINK) git-pbchk $(ROOTONBLDBIN)/git-nits

139 $(ROOTONBLDMAN1ONBLD)/git-nits.lonbld:
140     $(RM) $(ROOTONBLDMAN1ONBLD)/git-nits.lonbld
141     $(SYMLINK) git-pbchk.lonbld $(ROOTONBLDMAN1ONBLD)/git-nits.lonbld

143 install: all .WAIT $(ROOTONBLDSHFILES) $(ROOTONBLDPERLFILES) \
144             $(ROOTONBLDPERLMODULES) $(ROOTONBLDPYFILES) \
145             $(ROOTONBLDSRIPTLINKS) $(ROOTONBLDMAN1ONBLDFILES) \
146             $(ROOTONBLDMAKEFILES) $(ROOTONBLDETCFILES) \
147             $(ROOTONBLDEXCEPTFILES) $(ROOTONBLDMAN1ONBLDLINKS)

149 clean:
150     $(RM) $(CLEANFILES)

152 bldenv: bldenv.sh stdenv.sh
153     $(RM) "$@"
154     sed -e '/# STDENV_START/ r stdenv.sh' bldenv.sh > "$@"
155     # Check for shell lint and fail if we hit warnings
156     shlintout=$( $( /usr/bin/ksh93 -n "$@" 2>&1 ) ; \
157     [ " $$shlintout" != "" ] ) && \
158     { print -r -- " $$shlintout" ; false ; } || true
159     $(CHMOD) +x "$@"

161 bldenv.lonbld: bldenv
162     $(RM) "$@"
163     (set +o errexit ; ksh93 $? --nroff ; true) 2>&1 | \
164     sed -e 's/\.DS/.nf/g;s/\.DE/.fi/' \
165     -e 's/\.TH BLDENV 1/.TH BLDENV 1ONBLD/' \
166     -e 's/(1)/(1ONBLD)/' > "$@"

168 nightly: nightly.sh stdenv.sh
169     $(RM) "$@"
170     sed -e '/# STDENV_START/ r stdenv.sh' nightly.sh > nightly
171     $(CHMOD) +x "$@"

173 include ../Makefile targ
```

```
*****
15791 Fri Feb 2 23:18:40 2018
new/usr/src/tools/scripts/webrev.lonbld
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
*****
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 .\" Copyright 2010 Sun Microsystems, Inc. All rights reserved.
22 .\" Use is subject to license terms.
23 .\""
24 .\""
25 .TH WEBREV 1ONBLD "Mar 27, 2016"
26 .SH NAME
27 webrev \- Generate HTML codereview materials
28 .SH SYNOPSIS
29 .B webrev
30 [
31 .I common-options
32 ]
33 .B webrev
34 [ .I common-options
35 .I file-list-file
36 | .I -
37 ]
38 .B webrev
39 [
40 .I -
41 .B webrev
42 [
43 .I common-options
44 ]
45 .B -w
46 .I wx-file
47 .B webrev
48 .SH DESCRIPTION
49 .B webrev
50 builds a set of HTML files suitable for performing code review of
51 source changes in a web browser.
52 It supports Git and Subversion repositories.
53 It supports Mercurial, Git and Subversion repositories.
54 At its most basic, usage is:
55 .nf
56     $ webrev
57 .fi
```

```
59 In which case \fBwebrev\fR attempts to figure out the list of files
60 for review. If that fails, or if more control
61 over the set of files is needed, a \fIfile list\fR may be specified.
62 \fBwebrev\fR also attempts to deduce a
63 .I basis for comparison
64 (interchangeably called the \fIparent\fR, but see SCM INTERACTIONS below).
65 A basis for comparison is needed in order to determine the differences
66 introduced by the code changes under review.

67 By default, \fBwebrev\fR creates a \fIwebrev\fR directory in the
68 workspace directory that contains the generated HTML files, a generated
69 PDF review, and a patch representing the changes. It also places a
70 copy of the file list in that directory, and of both the old and new
71 raw files in the \fB$webrev_root/raw_files\fR directory.
72 To output the webrev somewhere other than the default location, use the
73 \fI-o <outdir\fR option, or set the \fBWDIR\fR environment variable.
74 For example:
75 .nf
76     $ webrev -o ~/public_html/myreview/
77 .fi
78 .PP
80 In the index file, each file is listed on a line with a link to the
81 relevant review materials. Comments for each change will be included
82 automatically. Cross references to bug (or other information) tracking
83 databases in the comments will become hyperlinks in the associated web
84 interface, according to the rules in CROSS REFERENCING below.

85 As a review aid, content may be added to the \fIindex\fR file in two ways.
86 First, the author may manually edit the file (for example by including
87 text that explains the changes in front of the links for each file).
88 Note that if webrev is run again, manual edits will be lost. Second,
89 if a file named \fIwebrev-info\fR is present at the root of the workspace,
90 it will be automatically included in the \fIindex\fR file. To include a
91 different file, see the \fI-i\fR option.

92 For each file in the file list, \fBwebrev\fR compares the file with the
93 version in the basis for comparison (i.e. the parent workspace) and
94 generates a variety of HTML renderings of the differences between
95 the two files; which of these renderings to use is largely a matter
96 of personal preference. Additional, webrev emits a patch, the old
97 and new versions of the file, and a "raw" copy of the file which is
98 suitable for download. For files which express differences, source
99 is formatted according to the following color coding:
100 .IP
101 .nf
102     unchanged : black
103     removed : brown
104     changed : blue
105     new : bold blue
106 .fi
107 .PP
108 .SH SCM INTERACTIONS
109 .PP
110 .B webrev
111 attempts to interact with the source code management system currently in use.
112 .B webrev
113 needs to be able locate the code under review (i.e. the workspace) and
114 the basis for comparison (i.e. the parent). The method for doing so
115 depends upon the SCM in use, which
116 .B webrev
117 will also attempt to auto-discover. In all cases,
118 .B webrev
119 must either discover the list of files which have changed, or else this list
120 must be manually specified, either in "webrev file list" format or in "wx"
121 format.
```

```

124 See FILE LIST for more details.
125 .PP
126 In all cases, if the user has activated the workspace with the
127 .BR ws (1ONBLD)
128 or
129 .BR bldenv (1ONBLD)
130 commands, \fBwebrev\fR will use the \fBCODEMGR_PARENT\fR and
131 \fBCODEMGR_WSV\fR environment variables to identify parent and child
132 workspaces respectively.
133 To manually specify the basis for comparison, use the -p option or
134 specify the \fBCODEMGR_PARENT\fR variable in either the file list or
135 the environment.

137 .SS Discovering the SCM in use.
138 .B webrev
139 makes use of
140 .BR which_scm (1ONBLD)
141 to determine the SCM in use for a given workspace.

143 .SS Mercurial
144 In the case of Mercurial \fBwebrev\fR will attempt to use the output
145 from the
146 .BR hg (1)
147 "hg root" command to identify the workspace root, and the
148 "hg path default" command to identify the parent workspace.

143 .SS Git
144 In the case of Git \fBwebrev\fR will attempt to use the output from the
145 .BR git (1)
146 "git rev-parse --git-dir" command to identify the workspace root, and will
147 attempt to use the remote branch which the current branch is tracking as the
148 parent, if none is specified 'origin/master' will be used.

150 The parent specified when using git is, in all cases, a git 'tree-ish' and
151 never an actual git repository, remote or otherwise. Anything specifiable to
152 git as a tree-ish should, similarly, be specifiable as a parent for webrev.
153 This includes branches, explicit revisions, reflog entries, etc. See
154 .BR git-rev-parse (1)

156 .SS Subversion
157 In the case of Subversion \fBwebrev\fR will attempt to use the output
158 from the
159 .BR svn (1)
160 "svn info" to find the workspace root and subversion repository URL.
161 .PP
162 The file list will be created from the output of the "svn status" command.

164 .SH CROSS REFERENCING
165 .PP
166 After extracting comments (see FILE LIST below),
167 .B webrev
168 will translate cross references into hyperlinks. By default, information
169 about available information tracking systems can be found in
170 /opt/onbld/etc/its.reg, and the specification of a local domain and
171 selection and prioritization of systems
172 in /opt/onbld/etc/its.conf. These file formats are self documenting. Also
173 see the -I and -C options below.
174 .SH OPTIONS
175 .TP 10
176 .BI "-c" revision
177 Generate webrev for single commit specified by \fIrevision\fR (git only).
178 .TP 10
179 .BI "-C" priority-file
180 In addition to the system default and an optional user-supplied ~/its.conf,
181 use the specified file to specify a local domain list and prioritize the list
182 of information tracking systems to be searched automatically when resolving cros

```

```

183 references.
184 .TP 10
185 .BI "-D"
186 Delete remote webrev via SFTP. Default remote host is \fIcr.opensolaris.org\fR,
187 default remote directory for removal is the same as workspace/repository
188 basename. Remote target can be overridden using -t option. If combined with
189 -U the deletion will be performed first. Also, if used together with -U
190 and the removal fails, no upload is done. Without -U option no webrev will
191 be generated, just like if -n option was used. The deletion is done by
192 moving the webrev to special directory in user's home directory. It is
193 expected that the remote host periodically runs a script which deletes
194 the contents of this directory. See the ENVIRONMENT VARIABLES section for
195 more details about this directory.
196 .TP 10
197 .BI "-h" head-revision
198 Specify the explicit head to generate webrev from (git only).
199 .TP 10
200 .BI "-I" information-file
201 Use the specified file to seed the list of information tracking systems.
202 .TP 10
203 .BI "-i" include-file
204 Include the specified file into the index.html file which is generated
205 as part of the webrev. This allows a snippet of XHTML to be added by
206 the webrev author. User content is contained by a <div> tag and
207 the markup should validate as XHTML 1.0 Transitional.
208 .TP 10
209 .BI "-N"
210 Suppress all comments from all output forms html, txt and pdf.
211 .TP 10
212 .BI "-n"
213 Do not generate webrev. Useful whenever only upload is needed.
214 .TP 10
215 .B -O
216 Enable \fIOpenSolaris\fR mode: information tracking system hyperlinks
217 are generated using the EXTERNAL_URL field from the specified its.reg entry,
218 instead of the default INTERNAL_URL domain field, and sources which appear in
219 \fIusr/closed\fR are automatically elided from the review.
220 .TP 10
221 .BI "-o" output-dir
222 Place output from running the script in the directory specified. If
223 specified, this option takes precedence over the WDIR environment variable.
224 .TP 10
225 .BI "-p" basis-of-comparison
226 Specify a basis of comparison meaningful for the SCM currently in use.
227 See SCM INTERACTIONS and INCREMENTAL REVIEWS.
228 .TP 10
229 .BI "-t" target
230 Upload target. Specified in form of URI identifier. For SCP/SFTP it is
231 \fIssh://user@remote_host:remote_dir\fR and for rsync it is
232 \fIrsync://user@remote_host:remote_dir\fR. This option can override the
233 -o option if the URI is fully specified. The target is relative to
234 the top level directory of the default sftp/rsync directory tree.
235 .TP 10
236 .BI "-U"
237 Upload the webrev. Default remote host is \fIcr.opensolaris.org\fR.
238 Default transport is rsync. If it fails, fallback to SCP/SFTP transport
239 is done.
240 .TP 10
241 .BI "-w" wx-file
242 Extract the file list from the wx "active" file specified. 'wx' uses
243 this mode when invoking webrev. The list is assumed to be in the
244 format expected by the \fIxw\fR package. See FILE LIST, below.

246 .SH FILE LIST
247 .PP
248 .B Webrev

```

```

249 needs to be told or to discover which files have changed in a
250 given workspace. By default,
251 .B webrev
252 will attempt to autodetect the
253 list of changed files by first consulting
254 .BR wx "(1)."
255 If this information is not available, webrev tries to consult the SCM (Source
256 Code Manager) currently in use. If that fails, the user must intervene by
257 specifying either a file list or additional options specific to the SCM in use.

259 .SS Webrev Format
260 A webrev formatted file list contains a list of all the files to
261 be included in the review with paths relative to the workspace
262 directory, e.g.
263 .IP
264 .nf
265 \f(CWusr/src/uts/common/fs/nfs/nfs_subr.c
266 usr/src/uts/common/fs/nfs/nfs_export.c
267 usr/src/cmd/fs.d/nfs/mountd/mountd.c
268 .fi
269 .PP
270 Include the paths of any files added, deleted, or modified.
271 You can keep this list of files in the webrev directory
272 that webrev creates in the workspace directory
273 (CODEMGR_WS).

275 If CODEMGR_WS is not set, it may be specified as an environment variable
276 within the file list, e.g.
277 .IP
278 .nf
279 \f(CWCODEMGR_WS=/home/brent/myws
280 usr/src/uts/common/fs/nfs/nfs_subr.c
281 usr/src/uts/common/fs/nfs/nfs_export.c
282 usr/src/cmd/fs.d/nfs/mountd/mountd.c
283 .fi
284 .PP
285 To compare the workspace against one other than the parent (see also
286 the -p option), include a CODEMGR_PARENT line in the file list, like:
287 .IP
288 .nf
289 \f(CWCODEMGR_WS=/home/brent/myws
290 CODEMGR_PARENT=/ws/onnv-gate
291 usr/src/uts/common/fs/nfs/nfs_subr.c
292 usr/src/uts/common/fs/nfs/nfs_export.c
293 usr/src/cmd/fs.d/nfs/mountd/mountd.c
294 .fi
295 .PP
296 Finally, run webrev with the name of the file containing the file list as an
297 argument, e.g.
298 .nf
299     $ webrev file.list
300 .fi
301 .PP
302 If "-" is supplied as the name of the file, then stdin will be used.

304 .SS wx Format
305 If the \fI-w\fR flag is specified then \fBwebrev\fR
306 will assume the file list is in the format expected by the "wx" package:
307 pathname lines alternating with SCCS comment lines separated by blank
308 lines, e.g.
309 .IP
310 .nf
311 \f(CWusr/src/uts/common/fs/nfs/nfs_subr.c

313 1206578 Fix spelling error in comment

```

```

315 usr/src/uts/common/fs/nfs/nfs_export.c
317 4039272 cstyle fixes
319 usr/src/cmd/fs.d/nfs/mountd/mountd.c
321 1927634 mountd daemon doesn't handle expletives
322 .fi

324 .SH INCREMENTAL REVIEWS
325 When conducting multiple rounds of code review, it may be desirable to
326 generate a webrev which represents the delta between reviews. In this
327 case, set the parent workspace to the path to the old webrev:
328 .IP
329 .nf
331 \f(CW$ webrev -o ~/public_html/myreview-rd2/ \\
332             -p ~/public_html/myreview/
333 .fi

335 .SH ENVIRONMENT VARIABLES
336 The following environment variables allow for customization of \fBwebrev\fR:
338 .PP
339 \fBCDIFFCMD\fR and \fBDIFFCMD\fR are used when generating Cdiffs and Udiffs
340 respectively; their default values are "diff -b -C 5" and "diff -b -U
341 5". To generate diffs with more (or less) than 5 lines of context or
342 with more (or less) strict whitespace handling, set one or both of
343 these variables in the user environment accordingly.
345 \fBWDIR\fR sets the output directory. It is functionally equivalent to
346 the \fI-o\fR option.

348 \fBWDIFF\fR specifies the command used to generate Wdiffs. Wdiff generates a
349 full unified context listing with line numbers where unchanged
350 sections of code may be expanded and collapsed. It also provides a
351 "split" feature that shows the same file in two HTML frames one above the
352 other. The default path for this script is
353 /ws/onnv-gate/public/bin/wdiff but WDIFF may be set to customize this
354 to use a more convenient location.

356 \fBWEBREV_TRASH_DIR\fR specifies alternative location of trash directory
357 for remote webrev deletion using the \fI-D\fR option. The directory is
358 relative to the top level directory of the default sftp/rsync directory tree.
359 The default value of this directory is ".trash".

361 .SH UPLOADING WEBREVS
362 A webrev can be uploaded to remote site using the -U option. To simply
363 generate new webrev and upload it to the default remote host use the following
364 command:
365 .IP
366 .nf
367 \f(CW$ webrev -U
368 .fi
369 .PP
370 This will generate the webrev to local directory named 'webrev' and upload it
371 to remote host with remote directory name equal to local workspace/repository
372 name. To change both local and remote directory name, -U can be combined with
373 -o option. The following command will store the webrev to local directory named
374 "foo.onnv" and upload it to the remote host with the same directory name:
375 .IP
376 .nf
377 \f(CW$ webrev -U -o $CODEMGR_WS/foo.onnv
378 .fi
379 .PP
380 If there is a need for manual change of the webrev before uploading,

```

```

381 -U can be combined with -n option so that first command will just generate
382 the webrev and the second command will upload it without generating it again:
383 .IP
384 .nf
385 \f(CW$ webrev
386 \f(CW$ webrev -n -U
387 .fi
388 .PP
389 For custom remote targets, -t option allows to specify all components:
390 .IP
391 .nf
392 \f(CW$ webrev -U -t \\
393     ssh://user@cr.opensolaris.org:foo/bar/bugfix.onnv
394 .fi
395 .PP
396 If the remote path is specified as absolute, \fBwebrev\fR will assume all the
397 directories are already created. If the path is relative, \fBwebrev\fR will
398 try to create all needed directories. This only works with SCP/SFTP transport.
399 .PP
400 By default, rsync transport will use SSH for transferring the data to remote
401 site. To specify custom username, use entry in SSH client configuration file,
402 for example:
403 .IP
404 .nf
405 \f(CWHost cr.opensolaris.org
406     Hostname cr.opensolaris.org
407     User vkotal
408 .fi

410 .SH DELETING WEBREVS
411 When deleting a webrev directory on remote site which has a different name
412 than the basename of local repository it is necessary to specify the output
413 option:
414 .IP
415 .nf
416 \f(CW$ webrev -Do webrev-foo.onnv
417 .fi
418 .PP
419 Otherwise \fBwebrev\fR will attempt to remove remote directory with the same
420 name as basename of the local repository.
421 .PP
422 For the nested directory case it is necessary to specify the full target:
423 .IP
424 .nf
425 \f(CW$ webrev -D -t \\
426     ssh://user@cr.opensolaris.org:foo/bar/bugfix.onnv
427 .fi
428 .PP
429 This will remove just the \fIbugfix.onnv\fR directory.

431 .SH SEE ALSO
439 .BR hg "(1)," [REDACTED]
432 .BR git "(1)," [REDACTED]
433 .BR ssh_config "(4)," [REDACTED]
434 .BR svn "(1)," [REDACTED]
435 .BR which_scm "(1ONBLD)" [REDACTED]

437 .SH ACKNOWLEDGEMENTS
438 Acknowledgements to Rob Thurlow, Mike Eisler, Lin Ling,
439 Rod Evans, Mike Kupfer, Greg Onufer, Glenn Skinner,
440 Oleg Larin, David Robinson, Matthew Cross, David L. Paktor,
441 Neal Gafter, John Beck, Darren Moffat, Norm Shulman, Bill Watson,
442 Pedro Rubio and Bill Shannon for valuable feedback and insight in
443 building webrev.

445 Have fun!

```

```

446 .br
447 .nf
448 Brent Callaghan 11/28/96
449 .fi

```

```
new/usr/src/tools/scripts/webrev.sh
```

```
*****
83609 Fri Feb 2 23:18:42 2018
new/usr/src/tools/scripts/webrev.sh
9001 cdm is useless, remove it
9002 webrev should know how to get the git user name
Reviewed by: Yuri Pankov <yuri.pankov@nexenta.com>
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
*****
_____ unchanged_portion_omitted _____
```

```
1046 #
1047 # fix_postscript
1048 #

1563 #
1564 # comments_from_wx {text|html} filepath
1565 #
1566 # Given the pathname of a file, find its location in a "wx" active
1567 # file list and print the following comment. Output is either text or
1568 # HTML; if the latter, embedded bugids (sequence of 5 or more digits)
1569 # are turned into URLs.
1570 #
1571 # This is also used with Mercurial and the file list provided by hg-active.
1572 #
1573 comments_from_wx()
1574 {
1575     typeset fmt=$1
1576     typeset p=$2
1577
1578     comm='$AWK '
1579     $1 == "'$p'" {
1580         do getline ; while (NF > 0)
1581             getline
1582             while (NF > 0) { print ; getline }
1583             exit
1584     } < $wxfile'
1585
1586     if [[ -z $comm ]]; then
1587         comm="*** NO COMMENTS ***"
1588
1589     if [[ $fmt == "text" ]]; then
1590         print -- "$comm"
1591     return
1592
1593     print -- "$comm" | html_quote | its2url
1594
1595 }

1596 #
1597 getcomments {text|html} filepath parentpath
1598 #
1599 #
1600 # Fetch the comments depending on what SCM mode we're in.
1601 #
1602 getcomments()
1603 {
1604     typeset fmt=$1
1605     typeset p=$2
1606     typeset pp=$3
1607
1608     if [[ -n $Nflag ]]; then
1609         return
1610     fi
```

```
1
```

```
new/usr/src/tools/scripts/webrev.sh
```

```
1613 #
1614 # Mercurial support uses a file list in wx format, so this
1615 # will be used there, too
1616 #
1617 if [[ -n $wxfile ]]; then
1618     comments_from_wx $fmt $p
1619 fi
1620
1621 _____ unchanged_portion_omitted ____

1791 #
1792 # Call hg-active to get the active list output in the wx active list format
1793 #
1794 function hg_active_wxfile
1795 {
1796     typeset child=$1
1797     typeset parent=$2
1798
1799 TMPFLIST=/tmp/$$.active
1800 $HG_ACTIVE -w $child -p $parent -o $TMPFLIST
1801 wxfile=$TMPFLIST
1802
1803
1804
1805
1806
1807 }

1808 #
1809 # flist_from_mercurial
1810 # Call hg-active to get a wx-style active list, and hand it off to
1811 # flist_from_wx
1812 #
1813 #
1814 function flist_from_mercurial
1815 {
1816     typeset child=$1
1817     typeset parent=$2
1818
1819 print " File list from: hg-active -p $parent ... \c"
1820 if [[ ! -x $HG_ACTIVE ]]; then
1821     print "" # Blank line for the \c above
1822     print -u2 "Error: hg-active tool not found. Exiting"
1823     exit 1
1824 fi
1825 hg_active_wxfile $child $parent
1826
1827 # flist_from_wx prints the Done, so we don't have to.
1828 flist_from_wx $TMPFLIST
1829 }

1830
1831 #
1832 # Transform a specified 'git log' output format into a wx-like active list.
1833 #
1834 function git_wxfile
1835 {
1836     typeset child="$1"
1837     typeset parent="$2"
1838
1839 TMPFLIST=/tmp/$$.active
1840 $PERL -e 'my (%files, %realfiles, $msg);
1841 my $parent = $ARGV[0];
1842 my $child = $ARGV[1];
1843
1844 open(F, "git diff -M --name-status $parent..$child | ");
1845 while (<F>) {
1846     chomp;
1847     if (/^R(\d+)\s+([^\s]+)\s+([^\s]+)/) { # rename
1848         if ($1 >= 75) { # Probably worth treating as a rename
1849             $realfiles{$3} = $2;
1850         } else {
1851             $realfiles{$3} = $1;
1852         }
1853     }
1854 }
```

```
2
```

```

1812             $realfiles{$2} = $2;
1813         }
1814     } else {
1815         my $f = (split /\s+/, $_)[1];
1816         $realfiles{$f} = $f;
1817     }
1818 }
1819 close(F);

1820 my $state = 1;           # 0|comments, 1|files
1821 open(F, "git whatchanged --pretty=format:%B $parent..$child |");
1822 while (<F>) {
1823     chomp;
1824     if (/^:[0-9]{6}/) {
1825         my ($unused, $fname, $fname2) = split(/\t/, $_);
1826         $fname = $fname2 if defined($fname2);
1827         next if !defined($realfiles{$fname}); # No real change
1828         $state = 1;
1829         chomp $msg;
1830         $files{$fname} .= $msg;
1831     } else {
1832         if ($state == 1) {
1833             $state = 0;
1834             $msg = /^n/ ? "" : "\n";
1835         }
1836         $msg .= "$_\n" if ($_);
1837     }
1838 }
1839 close(F);
1840
1841 for (sort keys %files) {
1842     if ($realfiles{$_} ne $_) {
1843         print "$_ $realfiles{$_}\n$files{$_}\n\n";
1844     } else {
1845         print "$_\n$files{$_}\n\n"
1846     }
1847 }
1848 } ${parent} ${child} > $TMPFLIST
1849
1850 wxfile=$TMPFLIST
1851 }



---



unchanged_portion omitted


1852 function build_old_new_mercurial
1853 {
1854     typeset olddir="$1"
1855     typeset newdir="$2"
1856     typeset old_mode=
1857     typeset new_mode=
1858     typeset file
1859
1860     #
1861     # Get old file mode, from the parent revision manifest entry.
1862     # Mercurial only stores a "file is executable" flag, but the
1863     # manifest will display an octal mode "644" or "755".
1864     #
1865     if [[ "$PDIR" == "." ]]; then
1866         file="$PF"
1867     else
1868         file="$PDIR/$PF"
1869     fi
1870     file='echo $file | $SED 's#/#\n#g'
1871     # match the exact filename, and return only the permission digits
1872     old_mode='$SED -n -e "/^\\(....\\) . ${file}\\$/s//\\1/p" \
1873         < $HG_PARENT_MANIFEST'
1874
1875     #

```

```

2005     # Get new file mode, directly from the filesystem.
2006     # Normalize the mode to match Mercurial's behavior.
2007     #
2008     new_mode='get_file_mode $CWS/$DIR/$F'
2009     if [[ -n "$new_mode" ]]; then
2010         if [[ "$new_mode" =~ *[1357]* ]]; then
2011             new_mode=755
2012         else
2013             new_mode=644
2014         fi
2015     fi
2016
2017     #
2018     # new version of the file.
2019     #
2020     rm -rf $newdir/$DIR/$F
2021     if [[ -e $CWS/$DIR/$F ]]; then
2022         cp $CWS/$DIR/$F $newdir/$DIR/$F
2023         if [[ -n $new_mode ]]; then
2024             chmod $new_mode $newdir/$DIR/$F
2025         else
2026             # should never happen
2027             print -u2 "ERROR: set mode of $newdir/$DIR/$F"
2028         fi
2029     fi
2030
2031     #
2032     # parent's version of the file
2033     #
2034     # Note that we get this from the last version common to both
2035     # ourselves and the parent. References are via $CWS since we have no
2036     # guarantee that the parent workspace is reachable via the filesystem.
2037     #
2038     if [[ -n $parent_webrev && -e $PWS/$PDIR/$PF ]]; then
2039         cp $PWS/$PDIR/$PF $olddir/$PDIR/$PF
2040     elif [[ -n $HG_PARENT ]]; then
2041         hg cat -R $CWS -r $HG_PARENT $CWS/$PDIR/$PF > \
2042             $olddir/$PDIR/$PF 2>/dev/null
2043
2044     if (( $? != 0 )); then
2045         rm -f $olddir/$PDIR/$PF
2046     else
2047         if [[ -n $old_mode ]]; then
2048             chmod $old_mode $olddir/$PDIR/$PF
2049         else
2050             # should never happen
2051             print -u2 "ERROR: set mode of $olddir/$PDIR/$PF"
2052         fi
2053     fi
2054 }
2055 }

1941 function build_old_new_git
1942 {
1943     typeset olddir="$1"
1944     typeset newdir="$2"
1945     typeset o_mode=
1946     typeset n_mode=
1947     typeset o_object=
1948     typeset n_object=
1949     typeset OWD=$PWD
1950     typeset file
1951     typeset type
1952
1953     cd $CWS

```

```

1955      #
1956      # Get old file and its mode from the git object tree
1957      #
1958      if [[ "$PDIR" == "." ]]; then
1959          file="$PF"
1960      else
1961          file="$PDIR/$PF"
1962      fi
1963
1964      if [[ -n $parent_webrev && -e $PWS/$PDIR/$PF ]]; then
1965          cp $PWS/$PDIR/$PF $olddir/$PDIR/$PF
1966      else
1967          $GIT ls-tree $GIT_PARENT $file | read o_mode type o_object junk
1968          $GIT cat-file $type $o_object > $olddir/$file 2>/dev/null
1969
1970          if (( $? != 0 )); then
1971              rm -f $olddir/$file
1972          elif [[ -n $o_mode ]]; then
1973              # Strip the first 3 digits, to get a regular octal mode
1974              o_mode=${o_mode:???}
1975              chmod $o_mode $olddir/$file
1976          else
1977              # should never happen
1978              print -u2 "ERROR: set mode of $olddir/$file"
1979          fi
1980      fi
1981
1982      #
1983      # new version of the file.
1984      #
1985      if [[ "$DIR" == "." ]]; then
1986          file="$F"
1987      else
1988          file="$DIR/$F"
1989      fi
1990      rm -rf $newdir/$file
1991
1992      if [[ -e $CWS/$DIR/$F ]]; then
1993          cp $CWS/$DIR/$F $newdir/$DIR/$F
1994          chmod $(get_file_mode $CWS/$DIR/$F) $newdir/$DIR/$F
1995      fi
1996      cd $PWD
1997 }
_____unchanged_portion_omitted_____
2039 function build_old_new
2040 {
2041     typeset WDIR=$1
2042     typeset PWS=$2
2043     typeset PDIR=$3
2044     typeset PF=$4
2045     typeset CWS=$5
2046     typeset DIR=$6
2047     typeset F=$7
2048
2049     typeset olddir="$WDIR/raw_files/old"
2050     typeset newdir="$WDIR/raw_files/new"
2051
2052     mkdir -p $olddir/$PDIR
2053     mkdir -p $newdir/$DIR
2054
2055     if [[ $SCM_MODE == "git" ]]; then
2056         if [[ $SCM_MODE == "mercurial" ]]; then
2057             build_old_new_mercurial "$olddir" "$newdir"
2058         elif [[ $SCM_MODE == "git" ]]; then
2059             build_old_new_git "$olddir" "$newdir"
2060         fi
2061
2062     fi
2063
2064     if [[ ! -f $olddir/$PDIR/$PF && ! -f $newdir/$DIR/$F ]]; then
2065         print "*** Error: file not in parent or child"
2066         return 1
2067     fi
2068 }
_____unchanged_portion_omitted_____

```

```

2057     elif [[ $SCM_MODE == "subversion" ]]; then
2058         build_old_new_subversion "$olddir" "$newdir"
2059     elif [[ $SCM_MODE == "unknown" ]]; then
2060         build_old_new_unknown "$olddir" "$newdir"
2061     fi
2062
2063     if [[ ! -f $olddir/$PDIR/$PF && ! -f $newdir/$DIR/$F ]]; then
2064         print "*** Error: file not in parent or child"
2065         return 1
2066     fi
2067     return 0
2068 }
_____unchanged_portion_omitted_____
2107 #
2108 #
2109 # Main program starts here
2110 #
2111 #
2113 trap "rm -f /tmp/$$.* ; exit" 0 1 2 3 15
2115 set +o noclobber
2117 PATH=$(dirname "$(whence $0)":$PATH
2119 [[ -z $WDIFF ]] && WDIFF='look_for_prog wdiff'
2120 [[ -z $WX ]] && WX='look_for_prog wx'
2121 [[ -z $HG_ACTIVE ]] && HG_ACTIVE='look_for_prog hg-active'
2122 [[ -z $GIT ]] && GIT='look_for_prog git'
2123 [[ -z $WHICH_SCM ]] && WHICH_SCM='look_for_prog which_scm'
2124 [[ -z $CODEREVIEW ]] && CODEREVIEW='look_for_prog codereview'
2125 [[ -z $PS2PDF ]] && PS2PDF='look_for_prog ps2pdf'
2126 [[ -z $PERL ]] && PERL='look_for_prog perl'
2127 [[ -z $RSYNC ]] && RSYNC='look_for_prog rsync'
2128 [[ -z $SSCCS ]] && SSCCS='look_for_prog sccs'
2129 [[ -z $AWK ]] && AWK='look_for_prog awk'
2130 [[ -z $GAWK ]] && GAWK='look_for_prog gawk'
2131 [[ -z $SCP ]] && SCP='look_for_prog scp'
2132 [[ -z $SED ]] && SED='look_for_prog sed'
2133 [[ -z $SFTP ]] && SFTP='look_for_prog sftp'
2134 [[ -z $SORT ]] && SORT='look_for_prog sort'
2135 [[ -z $MKTEMP ]] && MKTEMP='look_for_prog mktemp'
2136 [[ -z $GREP ]] && GREP='look_for_prog grep'
2137 [[ -z $FIND ]] && FIND='look_for_prog find'
2138 [[ -z $MANDOC ]] && MANDOC='look_for_prog mandoc'
2139 [[ -z $COL ]] && COL='look_for_prog col'
2141 # set name of trash directory for remote webrev deletion
2142 TRASH_DIR=".trash"
2143 [[ -n $WEBREV_TRASH_DIR ]] && TRASH_DIR=$WEBREV_TRASH_DIR
2145 if [[ ! -x $PERL ]]; then
2146     print -u2 "Error: No perl interpreter found. Exiting."
2147     exit 1
2148 fi
2150 if [[ ! -x $WHICH_SCM ]]; then
2151     print -u2 "Error: Could not find which_scm. Exiting."
2152     exit 1
2153 fi
2155 #
2156 # These aren't fatal, but we want to note them to the user.
2157 # We don't warn on the absence of 'wx' until later when we've

```

```
new/usr/src/tools/scripts/webrev.sh

2158 # determined that we actually need to try to invoke it.
2159 #
2160 [[ ! -x $CODEREVIEW ]] && print -u2 "WARNING: codereview(1) not found."
2161 [[ ! -x $PS2PDF ]] && print -u2 "WARNING: ps2pdf(1) not found."
2162 [[ ! -x $WDIFF ]] && print -u2 "WARNING: wdiff not found."

2164 # Declare global total counters.
2165 integer TOTL TINS TDEL TMOD TUNC

2167 # default remote host for upload/delete
2168 typeset -r DEFAULT_REMOTE_HOST="cr.opensolaris.org"
2169 # prefixes for upload targets
2170 typeset -r rsync_prefix="rsync://"
2171 typeset -r ssh_prefix="ssh://"

2173 cflag=
2174 Cflag=
2175 Dflag=
2176 flist_mode=
2177 flist_file=
2178 hflag=
2179 iflag=
2180 Iflag=
2181 lflag=
2182 Nflag=
2183 nflag=
2184 Oflag=
2185 oflag=
2186 pflag=
2187 tflag=
2188 uflag=
2189 Uflag=
2190 wflag=
2191 remote_target=

2312 #
2313 # NOTE: when adding/removing options it is necessary to sync the list
2314 # with usr/src/tools/onbld/hgext/cdm.py
2315 #
2316 while getopts "c:C:D:h:i:I:l:nN:o:O:p:t:U:w" opt
2317 do
2318     case $opt in
2319         c)      cflag=1
2320                 codemgr_head=$OPTARG
2321                 codemgr_parent=$OPTARG~;;
2322         C)      Cflag=1
2323                 ITSCONF=$OPTARG;;
2324         D)      Dflag=1;;
2325         h)      hflag=1
2326                 codemgr_head=$OPTARG;;
2327         i)      iflag=1
2328                 INCLUDE_FILE=$OPTARG;;
2329         I)      Iflag=1
2330                 ITSREG=$OPTARG;;
2331         N)      Nflag=1;;
2332         n)      nflag=1;;
2333         O)      Oflag=1;;
```

```
7 new/usr/src/tools/scripts/webrev.sh

2220         o)      oflag=1
2221             # Strip the trailing slash to correctly form remote target.
2222             WDIR=${OPTARG%/};;
2223
2224         p)      pflag=1
2225             codemgr_parent=$OPTARG;;
2226
2227         t)      tflag=1
2228             remote_target=$OPTARG;;
2229
2230         U)      Uflag=1;;
2231
2232         w)      wflag=1;;
2233
2234         ?)      usage;;
2235     esac
2236 done

2237 FLIST=/tmp/$$.flist

2238 if [[ -n $wflag && -n $lflag ]]; then
2239     usage
2240 fi

2241 # more sanity checking
2242 if [[ -n $nflag && -z $Uflag ]]; then
2243     print "it does not make sense to skip webrev generation" \
2244           "without -U"
2245     exit 1
2246 fi

2247 if [[ -n $tflag && -z $Uflag && -z $Dflag ]]; then
2248     echo "remote target has to be used only for upload or delete"
2249     exit 1
2250 fi

2251 #
2252 # For the invocation "webrev -n -U" with no other options, webrev will assume
2253 # that the webrev exists in ${CWS}/webrev, but will upload it using the name
2254 # ${basename ${CWS}}. So we need to get CWS set before we skip any remaining
2255 # logic.
2256 #
2257 $WHICH_SCM | read SCM_MODE junk || exit 1

2258 if [[ $SCM_MODE == "git" ]]; then
2259 if [[ $SCM_MODE == "mercurial" ]]; then
2260     #
2261     # Mercurial priorities:
2262     # 1. hg root from CODEMGR_WS environment variable
2263     # 1a. hg root from CODEMGR_WS/usr/closed if we're somewhere under
2264     #      usr/closed when we run webrev
2265     # 2. hg root from directory of invocation
2266     #
2267     if [[ ${PWD} == "usr/closed" ]]; then
2268         testparent=${CODEMGR_WS}/usr/closed
2269         # If we're in OpenSolaris mode, we enforce a minor policy:
2270         # help to make sure the reviewer doesn't accidentally publish
2271         # source which is under usr/closed
2272         if [[ -n "$Oflag" ]]; then
2273             print -u2 "OpenSolaris output not permitted with" \
2274                   "usr/closed changes"
2275             exit 1
2276         fi
2277     else
2278         testparent=${CODEMGR_WS}
2279     fi
2280 fi
```

```

2407     [[ -z $codemgr_ws && -n $testparent ]] && \
2408         codemgr_ws=$(hg root -R $testparent 2>/dev/null)
2409     [[ -z $codemgr_ws ]] && codemgr_ws=$(hg root 2>/dev/null)
2410     CWS=$codemgr_ws
2411 elif [[ $SCM_MODE == "git" ]]; then
2412     #
2413     # Git priorities:
2414     # 1. git rev-parse --git-dir from CODEMGR_WS environment variable
2415     # 2. git rev-parse --git-dir from directory of invocation
2416     #
2417     [[ -z $codemgr_ws && -n $CODEMGR_WS ]] && \
2418         codemgr_ws=$GIT --git-dir=$CODEMGR_WS/.git rev-parse --git-dir \
2419             2>/dev/null
2420     [[ -z $codemgr_ws ]] && \
2421         codemgr_ws=$($GIT rev-parse --git-dir 2>/dev/null)

2422 if [[ "$codemgr_ws" == ".git" ]]; then
2423     codemgr_ws="${PWD}/${codemgr_ws}"
2424 fi

2425 if [[ "$codemgr_ws" = */*.git" ]]; then
2426     codemgr_ws=$(dirname $codemgr_ws) # Lose the '/.git'
2427 fi
2428 CWS="$codemgr_ws"
2429 elif [[ $SCM_MODE == "subversion" ]]; then
2430     #
2431     # Subversion priorities:
2432     # 1. CODEMGR_WS from environment
2433     # 2. Relative path from current directory to SVN repository root
2434     #
2435     if [[ -n $CODEMGR_WS && -d $CODEMGR_WS/.svn ]]; then
2436         CWS=$CODEMGR_WS
2437     else
2438         svn info | while read line; do
2439             if [[ $line == "URL: *" ]]; then
2440                 url=${line#URL: }
2441             elif [[ $line == "Repository Root: *" ]]; then
2442                 repo=${line#Repository Root: }
2443             fi
2444         done
2445         rel=${url#$repo}
2446         CWS=${PWD%$rel}
2447     fi
2448 fi
2449
2450 # If no SCM has been determined, take either the environment setting
2451 # setting for CODEMGR_WS, or the current directory if that wasn't set.
2452 #
2453 if [[ -z ${CWS} ]]; then
2454     CWS=${CODEMGR_WS:-.}
2455 fi
2456
2457 # If the command line options indicate no webrev generation, either
2458 # explicitly (-n) or implicitly (-D but not -U), then there's a whole
2459 # ton of logic we can skip.
2460 #
2461 # Instead of increasing indentation, we intentionally leave this loop
2462 # body open here, and exit via break from multiple points within.
2463 # Search for DO_EVERYTHING below to find the break points and closure.
2464 #
2465 for do_everything in 1; do
2466
2467 # DO_EVERYTHING: break point

```

```

2326 if [[ -n $nflag || ( -z $Uflag && -n $Dflag ) ]]; then
2327     break
2328 fi

2329 #
2330 # If this manually set as the parent, and it appears to be an earlier webrev,
2331 # then note that fact and set the parent to the raw_files/new subdirectory.
2332 #
2333 #
2334 if [[ -n $pflag && -d $codemgr_parent/raw_files/new ]]; then
2335     parent_webrev=$(readlink -f "$codemgr_parent")
2336     codemgr_parent=$(readlink -f "$codemgr_parent/raw_files/new")
2337 fi

2338 if [[ -z $wflag && -z $lflag ]]; then
2339     shift $((OPTIND - 1))

2340     if [[ $1 == "-" ]]; then
2341         cat > $FLIST
2342         flist_mode="stdin"
2343         flist_done=1
2344         shift
2345     elif [[ -n $1 ]]; then
2346         if [[ ! -r $1 ]]; then
2347             print -u2 "$1: no such file or not readable"
2348             usage
2349         fi
2350         cat $1 > $FLIST
2351         flist_mode="file"
2352         flist_file=$1
2353         flist_done=1
2354         shift
2355     else
2356         flist_mode="auto"
2357     fi
2358
2359 fi

2360 fi

2361 #
2362 # Before we go on to further consider -l and -w, work out which SCM we think
2363 # is in use.
2364 #
2365 #
2366 case "$SCM_MODE" in
2367     git|subversion)
2368     mercurial/git/subversion)
2369     ;;
2370     unknown)
2371         if [[ $flist_mode == "auto" ]]; then
2372             print -u2 "Unable to determine SCM in use and file list not spec"
2373             print -u2 "See which_scm(1) for SCM detection information."
2374             exit 1
2375         ;;
2376     *)
2377         if [[ $flist_mode == "auto" ]]; then
2378             print -u2 "Unsupported SCM in use ($SCM_MODE) and file list not"
2379             exit 1
2380         ;;
2381     esac
2382 esac

2383 print -u2 "    SCM detected: $SCM_MODE"

2384 if [[ -n $wflag ]]; then
2385     #
2386     # If the -w is given then assume the file list is in Bonwick's "wx"
2387     # command format, i.e. pathname lines alternating with SCCS comment
2388     # lines with blank lines as separators. Use the SCCS comments later

```

```

2391      # in building the index.html file.
2392      #
2393      shift $((OPTIND - 1))
2394      wxfile=$1
2395      if [[ -z $wxfile && -n $CODEMGR_WS ]]; then
2396          if [[ -r $CODEMGR_WS/wx/active ]]; then
2397              wxfile=$CODEMGR_WS/wx/active
2398          fi
2399      fi
2400
2401      [[ -z $wxfile ]] && print -u2 "wx file not specified, and could not " \
2402          "be auto-detected (check \$CODEMGR_WS)" && exit 1
2403
2404      if [[ ! -r $wxfile ]]; then
2405          print -u2 "$wxfile: no such file or not readable"
2406          usage
2407      fi
2408
2409      print -u2 " File list from: wx 'active' file '$wxfile' ... \c"
2410      flist_from_wx $wxfile
2411      flist_done=1
2412      if [[ -n "$*" ]]; then
2413          shift
2414      fi
2415      elif [[ $flist_mode == "stdin" ]]; then
2416          print -u2 " File list from: standard input"
2417      elif [[ $flist_mode == "file" ]]; then
2418          print -u2 " File list from: $flist_file"
2419      fi
2420
2421      if [[ $# -gt 0 ]]; then
2422          print -u2 "WARNING: unused arguments: $*"
2423      fi
2424
2425      #
2426      # Before we entered the DO_EVERYTHING loop, we should have already set CWS
2427      # and CODEMGR_WS as needed. Here, we set the parent workspace.
2428      #
2429
2430      if [[ $SCM_MODE == "mercurial" ]]; then
2431          #
2432          # Parent can either be specified with -p
2433          # Specified with CODEMGR_PARENT in the environment
2434          # or taken from hg's default path.
2435          #
2436
2437      if [[ $SCM_MODE == "git" ]]; then
2438          if [[ -z $codemgr_parent && -n $CODEMGR_PARENT ]]; then
2439              codemgr_parent=$CODEMGR_PARENT
2440          fi
2441
2442          if [[ -z $codemgr_parent ]]; then
2443              codemgr_parent='hg path -R $codemgr_ws default 2>/dev/null'
2444          fi
2445
2446          PWS=$codemgr_parent
2447
2448          #
2449          # If the parent is a webrev, we want to do some things against
2450          # the natural workspace parent (file list, comments, etc)
2451          #
2452          if [[ -n $parent_webrev ]]; then
2453              real_parent=$(hg path -R $codemgr_ws default 2>/dev/null)
2454          else
2455              real_parent=$PWS
2456          fi

```

```

2603      #
2604      # If hg-active exists, then we run it. In the case of no explicit
2605      # flist given, we'll use it for our comments. In the case of an
2606      # explicit flist given we'll try to use it for comments for any
2607      # files mentioned in the flist.
2608      #
2609      if [[ -z $flist_done ]]; then
2610          flist_from_mercurial $CWS $real_parent
2611          flist_done=1
2612      fi
2613
2614      #
2615      # If we have a file list now, pull out any variables set
2616      # therein. We do this now (rather than when we possibly use
2617      # hg-active to find comments) to avoid stomping specifications
2618      # in the user-specified flist.
2619      #
2620      if [[ -n $flist_done ]]; then
2621          env_from_flist
2622      fi
2623
2624      #
2625      # Only call hg-active if we don't have a wx formatted file already
2626      #
2627      if [[ -x $HG_ACTIVE && -z $wxfile ]]; then
2628          print " Comments from: hg-active -p $real_parent ... \c"
2629          hg_active_wxfile $CWS $real_parent
2630          print " Done."
2631      fi
2632
2633      #
2634      # At this point we must have a wx flist either from hg-active,
2635      # or in general. Use it to try and find our parent revision,
2636      # if we don't have one.
2637      #
2638      if [[ -z $HG_PARENT ]]; then
2639          eval '$SED -e "s/#.*$//"' $wxfile | $GREP HG_PARENT=
2640      fi
2641
2642      #
2643      # If we still don't have a parent, we must have been given a
2644      # wx-style active list with no HG_PARENT specification, run
2645      # hg-active and pull an HG_PARENT out of it, ignore the rest.
2646      #
2647      if [[ -z $HG_PARENT && -x $HG_ACTIVE ]]; then
2648          $HG_ACTIVE -w $codemgr_ws -p $real_parent | \
2649          eval '$SED -e "s/#.*$//"' | $GREP HG_PARENT=
2650      elif [[ -z $HG_PARENT ]]; then
2651          print -u2 "Error: Cannot discover parent revision"
2652          exit 1
2653      fi
2654
2655      pnode=$(trim_digest $HG_PARENT)
2656      PRETTY_PWS="$PWS (at ${pnode})"
2657      cnode=$(hg parent -R $codemgr_ws --template '{node/short}' \
2658          2>/dev/null)
2659      PRETTY_CWS="$CWS (at ${cnode})"
2660      elif [[ $SCM_MODE == "git" ]]; then
2661          #
2662          # Check that "head" revision specified with -c or -h is sane
2663          if [[ -n $cflag || -n $hflag ]]; then
2664              head_rev=$(GIT rev-parse --verify --quiet "$codemgr_head")
2665              if [[ -z $head_rev ]]; then
2666                  print -u2 "Error: bad revision ${codemgr_head}"
2667                  exit 1
2668              fi
2669      fi

```

```

2436     if [[ -z $codemgr_head ]]; then
2437         codemgr_head="HEAD";
2438     fi
2439
2440     # Parent can either be specified with -p, or specified with
2441     # CODEMGR_PARENT in the environment.
2442     if [[ -z $codemgr_parent && -n $CODEMGR_PARENT ]]; then
2443         codemgr_parent=$CODEMGR_PARENT
2444     fi
2445
2446     # Try to figure out the parent based on the branch the current
2447     # branch is tracking, if we fail, use origin/master
2448     this_branch=$(git branch | awk '$1 == "*" { print $2 }')
2449     par_branch="origin/master"
2450
2451     # If we're not on a branch there's nothing we can do
2452     if [[ $this_branch != "(no branch)" ]]; then
2453         GIT_for-each-ref \
2454             --format='%(refname:short) %(upstream:short)' \
2455             refs/heads/ \
2456             while read local remote; do \
2457                 if [[ "$local" == "$this_branch" ]]; then \
2458                     par_branch="$remote" \
2459                 fi \
2460             done \
2461     fi
2462
2463     if [[ -z $codemgr_parent ]]; then
2464         codemgr_parent=$par_branch
2465     fi
2466     PWS=$codemgr_parent
2467
2468     #
2469     # If the parent is a webrev, we want to do some things against
2470     # the natural workspace parent (file list, comments, etc)
2471     #
2472     if [[ -n $parent_webrev ]]; then
2473         real_parent=$par_branch
2474     else
2475         real_parent=$PWS
2476     fi
2477
2478     if [[ -z $flist_done ]]; then
2479         flist_from_git "$codemgr_head" "$real_parent"
2480         flist_done=1
2481     fi
2482
2483     #
2484     # If we have a file list now, pull out any variables set
2485     # therein.
2486     #
2487     if [[ -n $flist_done ]]; then
2488         env_from_flist
2489     fi
2490
2491     #
2492     # If we don't have a wx-format file list, build one we can pull change
2493     # comments from.
2494     #
2495     if [[ -z $wxfile ]]; then
2496         print " Comments from: git...\c"
2497         git_wxfile "$codemgr_head" "$real_parent"
2498         print " Done."
2499     fi

```

```

2501     if [[ -z $GIT_PARENT ]]; then
2502         GIT_PARENT=$(git merge-base "$real_parent" "$codemgr_head")
2503     fi
2504     if [[ -z $GIT_PARENT ]]; then
2505         print -u2 "Error: Cannot discover parent revision"
2506         exit 1
2507     fi
2508
2509     pnode=$(trim_digest $GIT_PARENT)
2510
2511     if [[ -n $cflag ]]; then
2512         PRETTY_PWS="previous revision (at ${pnode})"
2513     elif [[ $real_parent == /*/* ]]; then
2514         origin=$(echo $real_parent | cut -d/ -f1)
2515         origin=$(git remote -v | \
2516             awk '$1 == "'$origin'" { print $2; exit }')
2517         PRETTY_PWS="${PWS} (${origin} at ${pnode})"
2518     elif [[ -n $pflag && -z $parent_webrev ]]; then
2519         PRETTY_PWS="${CWS} (explicit revision ${pnode})"
2520     else
2521         PRETTY_PWS="${PWS} (at ${pnode})"
2522     fi
2523
2524     cnode=$(git --git-dir=${codemgr_ws}/.git rev-parse --short=12 \
2525             ${codemgr_head} 2>/dev/null)
2526
2527     if [[ -n $cflag || -n $hflag ]]; then
2528         PRETTY_CWS="${CWS} (explicit head at ${cnode})"
2529     else
2530         PRETTY_CWS="${CWS} (at ${cnode})"
2531     fi
2532     elif [[ $SCM_MODE == "subversion" ]]; then
2533
2534         #
2535         # We only will have a real parent workspace in the case one
2536         # was specified (be it an older webrev, or another checkout).
2537         #
2538         [[ -n $codemgr_parent ]] && PWS=$codemgr_parent
2539
2540         if [[ -z $flist_done && $flist_mode == "auto" ]]; then
2541             flist_from_subversion $CWS $OLDPWD
2542         fi
2543     else
2544         if [[ $SCM_MODE == "unknown" ]]; then
2545             print -u2 "      Unknown type of SCM in use"
2546         else
2547             print -u2 "      Unsupported SCM in use: $SCM_MODE"
2548         fi
2549
2550     env_from_flist
2551
2552     if [[ -z $CODEMGR_WS ]]; then
2553         print -u2 "SCM not detected/supported and " \
2554             "$CODEMGR_WS not specified"
2555         exit 1
2556     fi
2557
2558     if [[ -z $CODEMGR_PARENT ]]; then
2559         print -u2 "SCM not detected/supported and " \
2560             "$CODEMGR_PARENT not specified"
2561         exit 1
2562     fi
2563
2564     CWS=$CODEMGR_WS
2565     PWS=$CODEMGR_PARENT
2566 fi

```

```

2568 #
2569 # If the user didn't specify a -i option, check to see if there is a
2570 # webrev-info file in the workspace directory.
2571 #
2572 if [[ -z $iflag && -r "$CWS/webrev-info" ]]; then
2573     iflag=1
2574     INCLUDE_FILE="$CWS/webrev-info"
2575 fi
2576
2577 if [[ -n $iflag ]]; then
2578     if [[ ! -r $INCLUDE_FILE ]]; then
2579         print -u2 "include file '$INCLUDE_FILE' does not exist or is" \
2580             "not readable."
2581         exit 1
2582     else
2583         #
2584         # $INCLUDE_FILE may be a relative path, and the script alters
2585         # PWD, so we just stash a copy in /tmp.
2586         #
2587         cp $INCLUDE_FILE /tmp/$$.include
2588     fi
2589 fi
2590
2591 # DO_EVERYTHING: break point
2592 if [[ -n $Nflag ]]; then
2593     break
2594 fi
2595
2596 typeset -A itsinfo
2597 typeset -r its_sed_script=/tmp/$$.its_sed
2598 valid_prefixes=
2599 if [[ -z $Nflag ]]; then
2600     REGFILE=$(dirname "$(whence $0)"/../etc/its.reg"
2601     if [[ -n $iflag ]]; then
2602         REGFILE=$ITSREG
2603     elif [[ -r ${HOME}/.its.reg ]]; then
2604         REGFILE=${HOME}/.its.reg
2605     else
2606         REGFILE=$DEFREGFILE
2607     fi
2608     if [[ ! -r $REGFILE ]]; then
2609         print "ERROR: Unable to read database registry file $REGFILE"
2610         exit 1
2611     elif [[ $REGFILE != $DEFREGFILE ]]; then
2612         print "    its.reg from: $REGFILE"
2613     fi
2614
2615     $SED -e '/^#/d' -e '/[^*]*$/d' $REGFILE | while read LINE; do
2616
2617         name=${LINE%%=*}
2618         value="${LINE##*=}"
2619
2620         if [[ $name == PREFIX ]]; then
2621             p=${value}
2622             valid_prefixes="$p ${valid_prefixes}"
2623         else
2624             itsinfo["${p}_${name}"]="$value"
2625         fi
2626     done
2627
2628     DEFCONFFILE=$(dirname "$(whence $0)"/../etc/its.conf"
2629     CONFFILES=$DEFCONFFILE
2630     if [[ -r ${HOME}/.its.conf ]]; then
2631         CONFFILES="$CONFFILES ${HOME}/.its.conf"
2632

```

```

2633         fi
2634         if [[ -n $Cflag ]]; then
2635             CONFFILES="${CONFFILES} ${ITSCONF}"
2636         fi
2637         its_domain=
2638         its_priority=
2639         for cf in ${CONFFILES}; do
2640             if [[ ! -r $cf ]]; then
2641                 print "ERROR: Unable to read database configuration file
2642                     $cf"
2643             elif [[ $cf != $DEFCONFFILE ]]; then
2644                 print "    its.conf: reading $cf"
2645             fi
2646             $SED -e '/^#/d' -e '/[^*]*$/d' $cf | while read LINE; do
2647                 eval "${LINE}"
2648             done
2649         done
2650
2651         #
2652         # If an information tracking system is explicitly identified by prefix,
2653         # we want to disregard the specified priorities and resolve it according
2654         #
2655         # To that end, we'll build a sed script to do each valid prefix in turn.
2656         #
2657         for p in ${valid_prefixes}; do
2658             #
2659             # When an informational URL was provided, translate it to a
2660             # hyperlink. When omitted, simply use the prefix text.
2661             #
2662             if [[ -z ${itsinfo["${p}_INFO"]} ]]; then
2663                 itsinfo["${p}_INFO"]=$p
2664             else
2665                 itsinfo["${p}_INFO"]="\${itsinfo\["\${p}\_INFO"\]}"
2666             fi
2667
2668             #
2669             # Assume that, for this invocation of webrev, all references
2670             # to this information tracking system should resolve through
2671             # the same URL.
2672             #
2673             # If the caller specified -O, then always use EXTERNAL_URL.
2674             #
2675             # Otherwise, look in the list of domains for a matching
2676             # INTERNAL_URL.
2677             #
2678             [[ -z $Oflag ]] && for d in ${its_domain}; do
2679                 if [[ -n ${itsinfo["${p}_INTERNAL_URL_${d}"]} ]]; then
2680                     itsinfo["${p}_URL"]="${itsinfo["${p}_INTERNAL_URL_${d}"]}"
2681                     break
2682                 fi
2683             done
2684             if [[ -z ${itsinfo["${p}_URL"]} ]]; then
2685                 itsinfo["${p}_URL"]="${itsinfo["${p}_EXTERNAL_URL"]}"
2686             fi
2687
2688             #
2689             # Turn the destination URL into a hyperlink
2690             #
2691             itsinfo["${p}_URL"]="\${itsinfo\["\${p}\_URL"\]}"
2692
2693             # The character class below contains a literal tab
2694             print "/^${p}: /{"
2695             s:${itsinfo["${p}_REGEX"]};${itsinfo["${p}_URL"]};g
2696             s:'${p}';${itsinfo["${p}_INFO"]};g
2697             }" >> ${its_sed_script}
2698         done

```

```

2700      #
2701      # The previous loop took care of explicit specification. Now use
2702      # the configured priorities to attempt implicit translations.
2703      #
2704      for p in ${its_priority}; do
2705          print "/^${itsinfo[$p}_REGEX}[^${itsinfo[$p}_REGEX}];${itsinfo[$p}_URL}";g
2706          s;^${itsinfo[$p}_REGEX}];${itsinfo[$p}_URL};;g
2707      } >> ${its_sed_script}
2708  done
2709 fi

2711 #
2712 # Search for DO_EVERYTHING above for matching "for" statement
2713 # and explanation of this terminator.
2714 #
2715 done

2717 #
2718 # Output directory.
2719 #
2720 WDIR=${WDIR:-$CWS/webrev}

2722 #
2723 # Name of the webrev, derived from the workspace name or output directory;
2724 # in the future this could potentially be an option.
2725 #
2726 if [[ -n $oflag ]]; then
2727     WNAME=${WDIR##*/}
2728 else
2729     WNAME=${CWS##*/}
2730 fi

2732 # Make sure remote target is well formed for remote upload/delete.
2733 if [[ -n $Dflag || -n $Uflag ]]; then
2734     #
2735     # If remote target is not specified, build it from scratch using
2736     # the default values.
2737     #
2738     if [[ -z $tflag ]]; then
2739         remote_target=${DEFAULT_REMOTE_HOST}:${WNAME}
2740     else
2741         #
2742         # Check upload target prefix first.
2743         #
2744         if [[ "${remote_target}" != ${rsync_prefix}* &&
2745             "${remote_target}" != ${ssh_prefix}* ]]; then
2746             print "ERROR: invalid prefix of upload URI" \
2747                 "($remote_target)"
2748             exit 1
2749         fi
2750         #
2751         # If destination specification is not in the form of
2752         # host_spec:remote_dir then assume it is just remote hostname
2753         # and append a colon and destination directory formed from
2754         # local webrev directory name.
2755         #
2756         typeset target_no_prefix=${remote_target##*://}
2757         if [[ $target_no_prefix == *:* ]]; then
2758             if [[ "${remote_target}" == *: ]]; then
2759                 remote_target=${remote_target}${WNAME}
2760             fi
2761         else
2762             if [[ ${target_no_prefix} == /* */ ]]; then
2763                 print "ERROR: badly formed upload URI" \
2764                     "($remote_target)"

```

```

2765                         exit 1
2766                     else
2767                         remote_target=${remote_target}: ${WNAME}
2768                     fi
2769                 fi
2770             fi
2771             #
2772             # Strip trailing slash. Each upload method will deal with directory
2773             # specification separately.
2774             #
2775             #
2776             remote_target=${remote_target%/>
2777         fi

2779 #
2780 # Option -D by itself (option -U not present) implies no webrev generation.
2781 #
2782 if [[ -z $Uflag && -n $Dflag ]]; then
2783     delete_webrev 1 1
2784     exit $?
2785 fi

2787 #
2788 # Do not generate the webrev, just upload it or delete it.
2789 #
2790 if [[ -n $nflag ]]; then
2791     if [[ -n $dflag ]]; then
2792         delete_webrev 1 1
2793         (( $? == 0 )) || exit $?
2794     fi
2795     if [[ -n $Uflag ]]; then
2796         upload_webrev
2797         exit $?
2798     fi
2799 fi

2801 if [ "${WDIR%/*}" ]; then
2802     WDIR=$PWD/$WDIR
2803 fi

2805 if [[ ! -d $WDIR ]]; then
2806     mkdir -p $WDIR
2807     (( $? != 0 )) && exit 1
2808 fi

2810 #
2811 # Summarize what we're going to do.
2812 #
2813 print "    Workspace: ${PRETTY_CWS:-$CWS}"
2814 if [[ -n $parent_webrev ]]; then
2815     print "Compare against: webrev at $parent_webrev"
2816 else
2817     print "Compare against: ${PRETTY_PWS:-$PWS}"
2818 fi

2820 [[ -n $INCLUDE_FILE ]] && print "    Including: $INCLUDE_FILE"
2821 print "        Output to: $WDIR"

2823 #
2824 # Save the file list in the webrev dir
2825 #
2826 [[ ! $FLIST -ef $WDIR/file.list ]] && cp $FLIST $WDIR/file.list

2828 rm -f $WDIR/${WNAME}.patch
2829 rm -f $WDIR/${WNAME}.ps
2830 rm -f $WDIR/${WNAME}.pdf

```

```

2832 touch $WDIR/$WNAME.patch
2834 print "    Output Files:"

2836 #
2837 # Clean up the file list: Remove comments, blank lines and env variables.
2838 #
2839 $SED -e "s/.*$/ / -e "/=d" -e "/^[_]*$/d" $FLIST > /tmp/$$.flist.clean
2840 FLIST=/tmp/$$.flist.clean

2842 #
2843 # For Mercurial, create a cache of manifest entries.
2844 #
2845 if [[ $SCM_MODE == "mercurial" ]]; then
2846     #
2847     # Transform the FLIST into a temporary sed script that matches
2848     # relevant entries in the Mercurial manifest as follows:
2849     # 1) The script will be used against the parent revision manifest,
2850     # so for FLIST lines that have two filenames (a renamed file)
2851     # keep only the old name.
2852     # 2) Escape all forward slashes the filename.
2853     # 3) Change the filename into another sed command that matches
2854     # that file in "hg manifest -v" output: start of line, three
2855     # octal digits for file permissions, space, a file type flag
2856     # character, space, the filename, end of line.
2857     # 4) Eliminate any duplicate entries. (This can occur if a
2858     # file has been used as the source of an hg cp and it's
2859     # also been modified in the same changeset.)
2860     #
2861     SEDFILE=/tmp/$$.manifest.sed
2862     $SED '
2863         s#^[^ ]* ###
2864         s#/#\ \/#g
2865         s#^.*#/.... . &$/##
2866     ' < $FLIST | $SORT -u > $SEDFILE

2867     #
2868     # Apply the generated script to the output of "hg manifest -v"
2869     # to get the relevant subset for this webrev.
2870     #
2871     HG_PARENT_MANIFEST=/tmp/$$.manifest
2872     hg -R $CWS manifest -v -r $HG_PARENT /
2873         $SED -n -f $SEDFILE > $HG_PARENT_MANIFEST
2874 fi

2875 # First pass through the files: generate the per-file webrev HTML-files.
2876 #
2877 cat $FLIST | while read LINE
2878 do
2879     set - $LINE
2880     P=$1

2881     #
2882     # Normally, each line in the file list is just a pathname of a
2883     # file that has been modified or created in the child. A file
2884     # that is renamed in the child workspace has two names on the
2885     # line: new name followed by the old name.
2886     #
2887     oldname=""
2888     oldpath=""
2889     rename=
2890     if [[ $# -eq 2 ]]; then
2891         PP=$2          # old filename
2892         if [[ -f $PP ]]; then

```

```

2893             oldname=" (copied from $PP)"
2894             else
2895                 oldname=" (renamed from $PP)"
2896             fi
2897             oldpath="$PP"
2898             rename=1
2899             PDIR=${PP%/*}
2900             if [[ $PDIR == $PP ]]; then
2901                 PDIR="."
2902             fi
2903             oldname=" (copied from $PP)"
2904             PDIR=$PP
2905             if [[ $DIR == $P ]]; then
2906                 DIR=".
2907             fi
2908             F=${P##*/}
2909             else
2910                 DIR=${P%/*}
2911                 if [[ "$DIR" == "$P" ]]; then
2912                     DIR=".
2913                 fi
2914             F=${P##*/}
2915             PP=$P
2916             PDIR=$DIR
2917             PF=$F
2918             fi
2919             COMM='getcomments html $P $PP'
2920             print "\t$P$oldname\n\t\t$c"
2921             #
2922             # Make the webrev mirror directory if necessary
2923             mkdir -p $WDIR/$DIR
2924             #
2925             # We stash old and new files into parallel directories in $WDIR
2926             # and do our diffs there. This makes it possible to generate
2927             # clean looking diffs which don't have absolute paths present.
2928             #
2929             build_old_new "$WDIR" "$PWS" "$PDIR" "$PF" "$CWS" "$DIR" "$F" || \
2930             continue
2931             #
2932             # Keep the old PWD around, so we can safely switch back after
2933             # diff generation, such that build_old_new runs in a
2934             # consistent environment.
2935             #
2936             OWD=$PWD
2937             cd $WDIR/raw_files
2938             #
2939             # The "git apply" command does not tolerate the spurious
2940             # "./" that we otherwise insert; be careful not to include
2941             # it in the paths that we pass to diff(1).
2942             #
2943             if [[ $PDIR == ". " ]]; then
2944                 ofile=old/$PF
2945             else
2946                 ofile=old/$PDIR/$PF

```

```

2928     fi
2929     if [[ $DIR == "." ]]; then
2930         nfile=new/$F
2931     else
2932         nfile=$DIR/$F
2933     fi
2934
2935     mv_but_nodiff=
2936     cmp $ofile $ofile > /dev/null 2>&1
2937     if [[ $? == 0 && $rename == 1 ]]; then
2938         mv_but_nodiff=1
2939     fi
2940
2941     #
2942     # If we have old and new versions of the file then run the appropriate
2943     # diffs. This is complicated by a couple of factors:
2944     #
2945     # - renames must be handled specially: we emit a 'remove'
2946     #   diff and an 'add' diff
2947     # - new files and deleted files must be handled specially
2948     # - GNU patch doesn't interpret the output of illumos diff
2949     #   properly when it comes to adds and deletes. We need to
2950     #   do some "cleansing" transformations:
2951     #       [to add a file] @@ -1,0 +X,Y @@ --> @@ -0,0 +X,Y @@
2952     #       [to del a file] @@ -X,Y +1,0 @@ --> @@ -X,Y +0,0 @@
2953     #
2954     cleanse_rmfile="$SED 's/^(@@ [0-9+,,-]* ) [0-9+,,-]* @@$/\1 +0,0 @@/'"
2955     cleanse_newfile="$SED 's/^@@ [0-9+,,-]* \\\([0-9+,,-]* @@\)$@@ -0,0 \1/'"
2956
2957     rm -f $WDIR/$DIR/$F.patch
2958     if [[ -z $rename ]]; then
2959         if [ ! -f "$ofile" ]; then
2960             diff -u /dev/null $ofile | sh -c "$cleanse_newfile" \
2961                 > $WDIR/$DIR/$F.patch
2962         elif [ ! -f "$ofile" ]; then
2963             diff -u $ofile /dev/null | sh -c "$cleanse_rmfile" \
2964                 > $WDIR/$DIR/$F.patch
2965         else
2966             diff -u $ofile $ofile > $WDIR/$DIR/$F.patch
2967         fi
2968     else
2969         diff -u $ofile /dev/null | sh -c "$cleanse_rmfile" \
2970                 > $WDIR/$DIR/$F.patch
2971
2972         diff -u /dev/null $ofile | sh -c "$cleanse_newfile" \
2973                 >> $WDIR/$DIR/$F.patch
2974     fi
2975
2976     #
2977     # Tack the patch we just made onto the accumulated patch for the
2978     # whole wad.
2979     #
2980     cat $WDIR/$DIR/$F.patch >> $WDIR/$WNAME.patch
2981     print " patch\c"
2982
2983     if [[ -f $ofile && -f $nfile && -z $mv_but_nodiff ]]; then
2984         ${CDIFFCMD:-diff -bt -C 5} $ofile $nfile > $WDIR/$DIR/$F.cdiff
2985         diff_to_html $F $DIR/$F "C" "$COMM" < $WDIR/$DIR/$F.cdiff \
2986             > $WDIR/$DIR/$F.cdiff.html
2987         print " cdiffs\c"
2988
2989         ${UDIFFCMD:-diff -bt -U 5} $ofile $nfile > $WDIR/$DIR/$F.udiff
2990         diff_to_html $F $DIR/$F "U" "$COMM" < $WDIR/$DIR/$F.udiff \
2991             > $WDIR/$DIR/$F.udiff.html
2992         print " udiffs\c"

```

```

2994
2995     if [[ -x $WDIFF ]]; then
2996         $WDIFF -c "$COMM" \
2997             -t "$WNAME Wdiff $DIR/$F" $ofile $nfile > \
2998                 $WDIR/$DIR/$F.wdiff.html 2>/dev/null
2999     if [[ $? -eq 0 ]]; then
3000         print " wdiffs\c"
3001     else
3002         print " wdiffs[fail]\c"
3003     fi
3004
3005     sdiff_to_html $ofile $nfile $F $DIR "$COMM" \
3006         > $WDIR/$DIR/$F.sdiff.html
3007     print " sdiffs\c"
3008     print " frames\c"
3009
3010     rm -f $WDIR/$DIR/$F.cdiff $WDIR/$DIR/$F.udiff
3011     difflines $ofile $nfile > $WDIR/$DIR/$F.count
3012     elif [[ -f $ofile && -f $nfile && -n $mv_but_nodiff ]]; then
3013         # renamed file: may also have differences
3014         difflines $ofile $nfile > $WDIR/$DIR/$F.count
3015     elif [[ -f $nfile ]]; then
3016         # new file: count added lines
3017         difflines /dev/null $nfile > $WDIR/$DIR/$F.count
3018     elif [[ -f $ofile ]]; then
3019         # old file: count deleted lines
3020         difflines $ofile /dev/null > $WDIR/$DIR/$F.count
3021     fi
3022
3023     #
3024     # Check if it's a man page, and create plain text, html and raw (ascii)
3025     # output for the new version, as well as diffs against old version.
3026     #
3027     if [[ -f "$ofile" && "$ofile" = *.+( [0-9]+ * ([a-zA-Z]) && \
3028         -x $MANDOC && -x $COL ]]; then
3029         $MANDOC -Tascii $ofile | $COL -b > $ofile.man.txt
3030         source_to_html txt < $ofile.man.txt > $ofile.man.txt.html
3031         print " man-txt\c"
3032         print "$MANCSS" > $WDIR/raw_files/new/$DIR/man.css
3033         $MANDOC -Thtml -Ostyle=man.css $ofile > $ofile.man.html
3034         print " man-html\c"
3035         $MANDOC -Tascii $ofile > $ofile.man.raw
3036         print " man-raw\c"
3037     if [[ -f "$ofile" && -z $mv_but_nodiff ]]; then
3038         $MANDOC -Tascii $ofile | $COL -b > $ofile.man.txt
3039         ${CDIFFCMD:-diff -bt -C 5} $ofile.man.txt \
3040             $ofile.man.txt > $WDIR/$DIR/$F.man.cdiff
3041         diff_to_html $F $DIR/$F "C" "$COMM" < \
3042             $WDIR/$DIR/$F.man.cdiff > \
3043             $WDIR/$DIR/$F.man.cdiff.html
3044         print " man-cdiffs\c"
3045         ${UDIFFCMD:-diff -bt -U 5} $ofile.man.txt \
3046             $ofile.man.txt > $WDIR/$DIR/$F.man.udiff
3047         diff_to_html $F $DIR/$F "U" "$COMM" < \
3048             $WDIR/$DIR/$F.man.udiff > \
3049             $WDIR/$DIR/$F.man.udiff.html
3050         print " man-udiffs\c"
3051     if [[ -x $WDIFF ]]; then
3052         $WDIFF -c "$COMM" -t "$WNAME Wdiff $DIR/$F" \
3053             $ofile.man.txt $nfile.man.txt > \
3054                 $WDIR/$DIR/$F.man.wdiff.html 2>/dev/null
3055     if [[ $? -eq 0 ]]; then
3056         print " man-wdiffs\c"
3057     else
3058         print " man-wdiffs[fail]\c"
3059     fi

```

```

3060         fi
3061             sdiff_to_html $ofile.man.txt $nfile.man.txt $F.man $DIR
3062                 "$COMM" > $WDIR/$DIR/$F.man.sdiff.html
3063             print " man-sdiffs\c"
3064             print " man-frames\c"
3065         fi
3066     rm -f $ofile.man.txt $nfile.man.txt
3067     rm -f $WDIR/$DIR/$F.man.cdiff $WDIR/$DIR/$F.man.udiff
3068 fi

3070 #
3071 # Now we generate the postscript for this file. We generate diffs
3072 # only in the event that there is delta, or the file is new (it seems
3073 # tree-killing to print out the contents of deleted files).
3074 #
3075 if [[ -f $nfile ]]; then
3076     ocr=$ofile
3077     [[ ! -f $ofile ]] && ocr=/dev/null

3079 if [[ -z $mv_but_nodiff ]]; then
3080     textcomm='getcomments text $P $PP'
3081     if [[ -x $CODEREVIEW ]]; then
3082         $CODEREVIEW -y "$textcomm" \
3083             -e $ocr $nfile \
3084             > /tmp/$$.psfile 2>/dev/null &&
3085             cat /tmp/$$.psfile >> $WDIR/$WNAME.ps
3086     if [[ $? -eq 0 ]]; then
3087         print " ps\c"
3088     else
3089         print " ps[fail]\c"
3090     fi
3091     fi
3092 fi

3095 if [[ -f $ofile ]]; then
3096     source_to_html Old $PP < $ofile > $WDIR/$DIR/$F-.html
3097     print " old\c"
3098 fi

3100 if [[ -f $nfile ]]; then
3101     source_to_html New $P < $nfile > $WDIR/$DIR/$F.html
3102     print " new\c"
3103 fi

3105 cd $OWD
3107 print
3108 done

3110 frame_nav_js > $WDIR/ancnav.js
3111 frame_navigation > $WDIR/ancnav.html

3113 if [[ ! -f $WDIR/$WNAME.ps ]]; then
3114     print " Generating PDF: Skipped: no output available"
3115 elif [[ -x $CODEREVIEW && -x $PS2PDF ]]; then
3116     print " Generating PDF: \c"
3117     fix_postscript $WDIR/$WNAME.ps | $PS2PDF - > $WDIR/$WNAME.pdf
3118     print "Done."
3119 else
3120     print " Generating PDF: Skipped: missing 'ps2pdf' or 'codereview'"
3121 fi

3123 # If we're in OpenSolaris mode and there's a closed dir under $WDIR,
3124 # delete it - prevent accidental publishing of closed source

```

```

3126 if [[ -n "$oflag" ]]; then
3127     $FIND $WDIR -type d -name closed -exec /bin/rm -rf {} \;
3128 fi

3130 # Now build the index.html file that contains
3131 # links to the source files and their diffs.

3133 cd $CWS

3135 # Save total changed lines for Code Inspection.
3136 print "$TOTL" > $WDIR/TotalChangedLines

3138 print " index.html: \c"
3139 INDEXFILE=$WDIR/index.html
3140 exec 3<&1                                # duplicate stdout to FD3.
3141 exec 1<&-                                # Close stdout.
3142 exec > $INDEXFILE                         # Open stdout to index file.

3144 print "$HTML<head>$STDHEAD"
3145 print "<title>$WNAME</title>"
3146 print "</head>"
3147 print "<body id=\"$UNWwebrev\">"
3148 print "<div class=\"summary\">"
3149 print "<h2>Code Review for $WNAME</h2>"

3151 print "<table>"

3153 #
3154 # Get the preparer's name:
3155 #
3156 # If the SCM detected is Git, and the configuration property user.name is
3157 # available, use that, but be careful to properly escape angle brackets (HTML
3158 # syntax characters) in the email address.
3159 # If the SCM detected is Mercurial, and the configuration property
3160 # ui.username is available, use that, but be careful to properly escape
3161 # angle brackets (HTML syntax characters) in the email address.
3162 #
3163 preparer=
3164 if [[ "$SCM_MODE" == git ]]; then
3165     preparer=$(git config user.name 2>/dev/null)
3166     if [[ "$SCM_MODE" == mercurial ]]; then
3167         preparer='hg showconfig ui.username 2>/dev/null'
3168     if [[ -n "$preparer" ]]; then
3169         preparer=$(echo "$preparer" | html_quote)"
3170     fi
3171     if [[ -z "$preparer" ]]; then
3172         preparer=$(
3173             $PERL -e '
3174                 ($login, $pw, $uid, $gid, $quota, $cmt, $gcos) = getpwuid($<);
3175                 if ($login) {
3176                     $gcos =~ s/^\&/ucfirst($login)/e;
3177                     printf "%s (%s)\n", $gcos, $login;
3178                 } else {
3179                     printf "(unknown)\n";
3180                 }
3181             ')
3183 PREPDATE=$(LC_ALL=C /usr/bin/date +%Y-%b-%d\ %R\ %z\ %Z)
3184 print "<tr><th>Prepared by:</th><td>$preparer on $PREPDATE</td></tr>"
3185 print "<tr><th>Workspace:</th><td>${PRETTY_CWS:-SCWS}</td></tr>"
3186 print "</td></tr>"
```

```

3187 print "<tr><th>Compare against:</th><td>"
3188 if [[ -n $parent_webrev ]]; then
3189     print "webrev at $parent_webrev"
3190 else
3191     print "${PRETTY_PWS:-$PWS}"
3192 fi
3193 print "</td></tr>" 
3194 print "<tr><th>Summary of changes:</th><td>" 
3195 printCI $TOTAL $TINS $TDEL $TMOD $TUNC
3196 print "</td></tr>" 

3198 if [[ -f $WDIR/$WNAME.patch ]]; then
3199     wpatch_url=$(print $WNAME.patch | url_encode)"
3200     print "<tr><th>Patch of changes:</th><td>" 
3201     print "<a href=\"$wpatch_url\">$WNAME.patch</a></td></tr>" 
3202 fi
3203 if [[ -f $WDIR/$WNAME.pdf ]]; then
3204     wpdf_url=$(print $WNAME.pdf | url_encode)"
3205     print "<tr><th>Printable review:</th><td>" 
3206     print "<a href=\"$wpdf_url\">$WNAME.pdf</a></td></tr>" 
3207 fi

3209 if [[ -n "$iflag" ]]; then
3210     print "<tr><th>Author comments:</th><td><div>" 
3211     cat /tmp/$$.include
3212     print "</div></td></tr>" 
3213 fi
3214 print "</table>" 
3215 print "</div>" 

3217 #
3218 # Second pass through the files: generate the rest of the index file
3219 #
3220 cat $FLIST | while read LINE
3221 do
3222     set - $LINE
3223     P=$1

3225     if [[ $# == 2 ]]; then
3226         PP=$2
3227         oldname="$PP"
3228     else
3229         PP=$P
3230         oldname=""
3231     fi

3233     mv_but_nodiff=
3234     cmp $WDIR/raw_files/old/$PP $WDIR/raw_files/new/$P > /dev/null 2>&1
3235     if [[ $? == 0 && -n "$oldname" ]]; then
3236         mv_but_nodiff=1
3237     fi

3239     DIR=${P%/*}
3240     if [[ $DIR == $P ]]; then
3241         DIR=.". "# File at root of workspace
3242     fi

3244     # Avoid processing the same file twice.
3245     # It's possible for renamed files to
3246     # appear twice in the file list

3248     F=$WDIR/$P
3250     print "<p>" 
3252     # If there's a diffs file, make diffs links

```

```

3254     if [[ -f $F.cdiff.html ]]; then
3255         cdiff_url=$(print $P.cdiff.html | url_encode)"
3256         udiff_url=$(print $P.udiff.html | url_encode)"
3257         sdiff_url=$(print $P.sdiff.html | url_encode)"
3258         frames_url=$(print $P.frames.html | url_encode)"
3259         print "<a href=\"$cdiff_url\">Cdiffs</a>" 
3260         print "<a href=\"$udiff_url\">Udiffs</a>" 
3261         if [[ -f $F.wdiff.html && -x $WDIFF ]]; then
3262             wdiff_url=$(print $P.wdiff.html | url_encode)"
3263             print "<a href=\"$wdiff_url\">Wdiffs</a>" 
3264         fi
3265         print "<a href=\"$sdiff_url\">Sdiffs</a>" 
3266         print "<a href=\"$frames_url\">Frames</a>" 
3267     else
3268         print "-----"
3269         if [[ -x $WDIFF ]]; then
3270             print "-----"
3271         fi
3272         print "-----"
3273     fi

3275     # If there's an old file, make the link
3277     if [[ -f $F-.html ]]; then
3278         oldfile_url=$(print $P-.html | url_encode)"
3279         print "<a href=\"$oldfile_url\">Old</a>" 
3280     else
3281         print "---"
3282     fi

3284     # If there's an new file, make the link
3286     if [[ -f $F.html ]]; then
3287         newfile_url=$(print $P.html | url_encode)"
3288         print "<a href=\"$newfile_url\">New</a>" 
3289     else
3290         print "---"
3291     fi

3293     if [[ -f $F.patch ]]; then
3294         patch_url=$(print $P.patch | url_encode)"
3295         print "<a href=\"$patch_url\">Patch</a>" 
3296     else
3297         print "---"
3298     fi

3300     if [[ -f $WDIR/raw_files/new/$P ]]; then
3301         rawfiles_url=$(print raw_files/new/$P | url_encode)"
3302         print "<a href=\"$rawfiles_url\">Raw</a>" 
3303     else
3304         print "---"
3305     fi

3307     print "<b>$P</b>" 
3309     # For renamed files, clearly state whether or not they are modified
3310     if [[ -f "$oldname" ]]; then
3311         if [[ -n "$mv_but_nodiff" ]]; then
3312             print "<i>(copied from $oldname)</i>" 
3313         else
3314             print "<i>(copied and modified from $oldname)</i>" 
3315         fi
3316     elif [[ -n "$oldname" ]]; then
3317         if [[ -n "$mv_but_nodiff" ]]; then
3318             print "<i>(renamed from $oldname)</i>" 

```

```

3319         else
3320             print "<i>(renamed and modified from $oldname)</i>"
3321         fi
3322     fi
3323
3324     # If there's an old file, but no new file, the file was deleted
3325     if [[ -f $F-.html && ! -f $F.html ]]; then
3326         print " <i>(deleted)</i>""
3327     fi
3328
3329     # Check for usr/closed and deleted_files/usr/closed
3330     if [ ! -z "$oflag" ]; then
3331         if [[ $P == usr/closed/* || \
3332             $P == deleted_files/usr/closed/* ]]; then
3333             print "&nbsp;&nbsp;<i>Closed source: omitted from" \
3334             "this review</i>""
3335         fi
3336     fi
3337
3338     manpage=
3339     if [[ -f $F.man.cdiff.html || \
3340         -f $WDIR/raw_files/new/$P.man.txt.html ]]; then
3341         manpage=1
3342         print "<br/>man:"
3343     fi
3344
3345     if [[ -f $F.man.cdiff.html ]]; then
3346         mancdiff_url=$(print $P.man.cdiff.html | url_encode)
3347         manudiff_url=$(print $P.man.udiff.html | url_encode)
3348         mansdiff_url=$(print $P.man.sdiff.html | url_encode)
3349         manframes_url=$(print $P.man.frames.html | url_encode)
3350         print "<a href=\"$mancdiff_url\">cdiffs</a>""
3351         print "<a href=\"$manudiff_url\">Udiffs</a>""
3352         if [[ -f $F.man.wdiff.html && -x $WDIFF ]]; then
3353             manwdiff_url=$(print $P.man.wdiff.html | url_encode)
3354             print "<a href=\"$manwdiff_url\">Wdiffs</a>""
3355         fi
3356         print "<a href=\"$mansdiff_url\">Sdiffs</a>""
3357         print "<a href=\"$manframes_url\">Frames</a>""
3358     elif [[ -n $manpage ]]; then
3359         print " ----- -----"
3360         if [[ -x $WDIFF ]]; then
3361             print " -----"
3362         fi
3363         print " ----- -----"
3364     fi
3365
3366     if [[ -f $WDIR/raw_files/new/$P.man.txt.html ]]; then
3367         mantxt_url=$(print raw_files/new/$P.man.txt.html | url_encode)
3368         print "<a href=\"$mantxt_url\">TXT</a>""
3369         manhtml_url=$(print raw_files/new/$P.man.html | url_encode)
3370         print "<a href=\"$manhtml_url\">HTML</a>""
3371         manraw_url=$(print raw_files/new/$P.man.raw | url_encode)
3372         print "<a href=\"$manraw_url\">Raw</a>""
3373     elif [[ -n $manpage ]]; then
3374         print " --- ---- ---"
3375     fi
3376
3377     print "</p>""
3378
3379     # Insert delta comments
3380     print "<blockquote><pre>""
3381     getcomments html $P $PP
3382     print "</pre>""
3383
3384     # Add additional comments comment

```

```

3385         print "<!-- Add comments to explain changes in $P here -->""
3386
3387         # Add count of changes.
3388         if [[ -f $F.count ]]; then
3389             cat $F.count
3390             rm $F.count
3391         fi
3392
3393         if [[ $SCM_MODE == "unknown" ]]; then
3394             if [[ $SCM_MODE == "mercurial" || \
3395                 $SCM_MODE == "unknown" ]]; then
3396                 # Include warnings for important file mode situations:
3397                 # 1) New executable files
3398                 # 2) Permission changes of any kind
3399                 # 3) Existing executable files
3400                 old_mode=
3401                 if [[ -f $WDIR/raw_files/old/$PP ]]; then
3402                     old_mode='get_file_mode $WDIR/raw_files/old/$PP'
3403
3404                 new_mode=
3405                 if [[ -f $WDIR/raw_files/new/$P ]]; then
3406                     new_mode='get_file_mode $WDIR/raw_files/new/$P'
3407
3408                 if [[ -z "$old_mode" && "$new_mode" = *[1357]* ]]; then
3409                     print "<span class=\"chmod\">>""
3410                     print "<p>new executable file: mode $new_mode</p>""
3411                     print "</span>""
3412                 elif [[ -n "$old_mode" && -n "$new_mode" && \
3413                     "$old_mode" != "$new_mode" ]]; then
3414                     print "<span class=\"chmod\">>""
3415                     print "<p>mode change: $old_mode to $new_mode</p>""
3416                     print "</span>""
3417                 elif [[ "$new_mode" = *[1357]* ]]; then
3418                     print "<span class=\"chmod\">>""
3419                     print "<p>executable file: mode $new_mode</p>""
3420                     print "</span>""
3421                 fi
3422
3423                 print "</blockquote>""
3424             done
3425
3426             print
3427             print
3428             print
3429             print "<hr></hr>""
3430             print "<p style=\"font-size: small\">>""
3431             print "This code review page was prepared using <b>$0</b>."
3432             print "Webrev is maintained by the <a href=\"http://www.illumos.org\">""
3433             print "illumos</a> project. The latest version may be obtained"
3434             print "<a href=\"http://src.illumos.org/source/xref/illumos-gate/usr/src/tools/s"
3435             print "</body>""
3436             print "</html>""
3437
3438             exec 1<&-                                # Close FD 1.
3439             exec 1<&3                                # dup FD 3 to restore stdout.
3440             exec 3<&-                                # close FD 3.
3441
3442             print "Done."
3443
3444             #
3445             # If remote deletion was specified and fails do not continue.
3446             #
3447             if [[ -n $Dflag ]]; then
3448                 delete_webrev 1 1

```

```
new/usr/src/tools/scripts/webrev.sh  
3449         (( $? == 0 )) || exit $?  
3450 fi  
  
3452 if [[ -n $Uflag ]]; then  
3453     upload_webrev  
3454     exit $?  
3455 fi
```

29