

```

*****
10877 Tue Apr 22 07:25:47 2014
new/usr/src/cmd/Makefile
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 # Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
23 # Copyright 2010 Nexenta Systems, Inc. All rights reserved.
24 # Copyright (c) 2012 Joyent, Inc. All rights reserved.
25 # Copyright (c) 2012 by Delphix. All rights reserved.
26 # Copyright (c) 2013 DEY Storage Systems, Inc. All rights reserved.
27 # Copyright 2014 Andrew Stormont.
28 #endif /* ! codereview */

30 include ../Makefile.master

32 #
33 # Note that the commands 'agents', 'lp', 'perl', and 'man' are first in
34 # the list, violating alphabetical order. This is because they are very
35 # long-running and should be given the most wall-clock time for a
36 # parallel build.
37 #
38 # Commands in the FIRST_SUBDIRS list are built before starting the build
39 # of other commands. Currently this includes only 'isaexec' and
40 # 'platexec'. This is necessary because $(ROOT)/usr/lib/isaexec or
41 # $(ROOT)/usr/lib/platexec must exist when some other commands are built
42 # because their 'make install' creates a hard link to one of them.
43 #
44 # Commands are listed one per line so that TeamWare can auto-merge most
45 # changes.
46 #

48 FIRST_SUBDIRS=      \
49     isaexec         \
50     platexec

52 COMMON_SUBDIRS=    \
53     allocate        \
54     availdevs       \
55     lp               \
56     perl             \
57     man              \
58     Adm              \
59     abi

```

```

60     adbgen           \
61     acct             \
62     acctadm          \
63     arch             \
64     asa              \
65     ast              \
66     audio            \
67     auths           \
68     autopush        \
69     avs              \
70     awk              \
71     awk_xpg4        \
72     backup           \
73     banner           \
74     bart             \
75     basename        \
76     bc               \
77     bdiff            \
78     beadm            \
79     bfs              \
80     bnu              \
81     boot             \
82     busstat         \
83     cal              \
84     calendar        \
85     captinfo        \
86     cat              \
87     cdrw             \
88     cfgadm           \
89     checkeq         \
90     checknr         \
91     chgrp            \
92     chmod            \
93     chown            \
94     chroot           \
95     clear            \
96     clinfo           \
97     cmd-crypto      \
98     cmd-inet        \
99     col              \
100    compress         \
101    consadm          \
102    coreadm          \
103    cpio             \
104    cpc              \
105    cron             \
106    crypt            \
107    csh              \
108    csplit           \
109    ctrun            \
110    ctstat           \
111    ctwatch          \
112    datadm           \
113    date             \
114    dc                \
115    dd               \
116    deroff           \
117    devfsadm        \
118    syseventd       \
119    devctl           \
120    devinfo          \
121    devmgmt          \
122    devprop          \
123    dfs.cmds         \
124    diff             \
125    diff3

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

455     ypcmd       \
456     yppasswd    \
457     zdb         \
458     zdump       \
459     zfs         \
460     zhack       \
461     zic         \
462     zinject     \
463     zlogin      \
464     zoneadm     \
465     zoneadmd    \
466     zonecfg     \
467     zonename    \
468     zpool       \
469     zlook       \
470     zonestat    \
471     zstreamdump \
472     ztest       \
473     \
474     i386_SUBDIRS= \
475     acpihp       \
476     addbadsec    \
477     biosdev      \
478     diskscan     \
479     lms           \
480     ntfsprogs    \
481     parted       \
482     rtc          \
483     ucodeadm     \
484     xvm          \
485     \
486     sparc_SUBDIRS= \
487     cvcd         \
488     dcs          \
489     device_remap \
490     drd          \
491     fruadm       \
492     ldmad        \
493     oplhpd       \
494     prtdscp      \
495     prtfru       \
496     scadm        \
497     sckmd        \
498     sf880drd    \
499     virtinfo     \
500     vntsd        \
501     \
502     #
503     # Commands that are messaged. Note that 'lp' and 'man' come first
504     # (see previous comment about 'lp' and 'man').
505     #
506     MSGSUBDIRS= \
507     lp           \
508     man          \
509     abi          \
510     acctadm     \
511     allocate     \
512     asa         \
513     audio       \
514     audit       \
515     auditconfig \
516     auditd      \
517     auditrecord \
518     auditset    \
519     auths       \
520     autopush    \

```

new/usr/src/cmd/Makefile

```

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

```

9

new/usr/src/cmd/Makefile

```

587     getconf
588     getent
589     gettext
590     gettxt
591     grep
592     grep_xpg4
593     grpck
594     gss
595     halt
596     head
597     hostname
598     hotplug
599     id
600     idmap
601     isaexec
602     iscsiadm
603     iscsid
604     isns
605     itadm
606     kbd
607     krb5
608     ksh
609     last
610     ldap
611     ldapcachemgr
612     lgrpinfo
613     locale
614     lofiadm
615     logadm
616     logger
617     logins
618     ls
619     luxadm
620     lvm
621     mailx
622     msg
623     mkdir
624     mkpwdict
625     mktemp
626     more
627     mpathadm
628     msgfmt
629     mv
630     ndmpadm
631     ndmpstat
632     newgrp
633     newtask
634     nice
635     nohup
636     oawk
637     pack
638     passwd
639     passmgmt
640     pathchk
641     pfexec
642     pg
643     pgrep
644     picl
645     pools
646     power
647     pr
648     praudit
649     print
650     profiles
651     projadd
652     projects

```

10

```

653 prstat //
654 prtdiag //
655 ps //
656 psrinfo //
657 ptools //
658 pwconv //
659 pwd //
660 pyzfs //
661 raidctl //
662 ramdiskadm //
663 rcap //
664 rcm_daemon //
665 refer //
666 regcmp //
667 renice //
668 roles //
669 rm //
670 rmdir //
671 rmformat //
672 rmmount //
673 rmvolmgr //
674 sasinfo //
675 sbdadm //
676 scadm //
677 script //
678 scsi //
679 sdiff //
680 sdpadm //
681 sgs //
682 sh //
683 shcomp //
684 smbstrv //
685 sort //
686 split //
687 srptadm //
688 ssh //
689 stat //
690 stmfadm //
691 stmsboot //
692 strings //
693 su //
694 svc //
695 svr4pkg //
696 swap //
697 syseventadm //
698 syseventd //
699 tabs //
700 tar //
701 tbl //
702 time //
703 tnf //
704 touch //
705 tput //
706 troff //
707 tsol //
708 tty //
709 ttymon //
710 tzreload //
711 ul //
712 uname //
713 units //
714 unlink //
715 unpack //
716 userattr //
717 valtools //
718 vgrind //

```

```

719 vi //
720 volcheck //
308 volrmmount //
721 vrrpadm //
722 vscan //
723 w //
724 who //
725 whodo //
726 wracct //
727 write //
728 wusbadm //
729 xargs //
730 yppasswd //
731 zdump //
732 zfs //
733 zic //
734 zlogin //
735 zoneadm //
736 zoneadm //
737 zonecfg //
738 zonename //
739 zpool //
740 zonestat //

742 sparc_MSGSUBDIRS= //
743 fruadm //
744 prtdscp //
745 prtfu //
746 virtinfo //
747 vntsd //

749 i386_MSGSUBDIRS= \
750 ucodeadm //

752 # //
753 # commands that use dcgettext for localized time, LC_TIME //
754 # //
755 DCSUBDIRS= //
756 cal //
757 cfgadm //
758 diff //
759 ls //
760 pr //
761 ps //
762 tar //
763 w //
764 who //
765 whodo //
766 write //

768 # //
769 # commands that belong only to audit. //
770 # //
771 AUDITSUBDIRS= //
772 amt //
773 audit //
774 audit_warn //
775 auditconfig //
776 auditd //
777 auditrecord //
778 auditreduce //
779 auditset //
780 auditstat //
781 praudit //

783 # //

```

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

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

797 .KEEP_STATE:

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

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

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

806 #
807 # Manifests cannot be checked in parallel, because we are using
808 # the global repository that is in $(SRC)/cmd/svc/seed/global.db.
809 # For this reason, to avoid .PARALLEL and .NO_PARALLEL conflicts,
810 # we spawn off a sub-make to perform the non-parallel 'make check'
811 #
812 check:
813     $(MAKE) -f Makefile.check check

815 #
816 # The .WAIT directive works around an apparent bug in parallel make.
817 # Evidently make was getting the target _msg vs. _dc confused under
818 # some level of parallelization, causing some of the _dc objects
819 # not to be built.
820 #
821 _msg: $(MSGSUBDIRS) $(MACH)_MSGSUBDIRS .WAIT _dc

823 _dc: $(DCSUBDIRS)

825 #
826 # Dependencies
827 #
828 fs.d: fstyp
829 ksh:  shcomp isaexec
830 mdb:  terminfo
831 print: lp

833 $(FIRST_SUBDIRS) $(BWOSDIRS) $(SUBDIRS) $(AUDITSUBDIRS): FRC
834     @if [ -f $@/Makefile ]; then \
835         cd $@; pwd; $(MAKE) $(TARGET); \
836     else \
837         true; \
838     fi

840 FRC:
```

new/usr/src/cmd/rmformat/Makefile

1

1589 Tue Apr 22 07:25:48 2014

new/usr/src/cmd/rmformat/Makefile

4833 Remove volrmmount

Reviewed by: Dan McDonald <danmcd@omniti.com>

Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>

```
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, Version 1.0 only
6 # (the "License"). You may not use this file except in compliance
7 # with the License.
8 #
9 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10 # or http://www.opensolaris.org/os/licensing.
11 # See the License for the specific language governing permissions
12 # and limitations under the License.
13 #
14 # When distributing Covered Code, include this CDDL HEADER in each
15 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16 # If applicable, add the following below this CDDL HEADER, with the
17 # fields enclosed by brackets "[]" replaced with your own identifying
18 # information: Portions Copyright [yyyy] [name of copyright owner]
19 #
20 # CDDL HEADER END
21 #
22 #
23 # Copyright 2005 Sun Microsystems, Inc. All rights reserved.
24 # Use is subject to license terms.
25 #
26 # Copyright 2014 Andrew Stormont.
27 #
28 #endif /* ! codereview */

30 PROG= rmformat

32 OBJS=  rmf_main.o rmf_menu.o rmf_misc.o rmf_slice.o

34 include ../Makefile.cmd

36 SRCS= $(OBJS:.o=.c)

38 LDLIBS +=      -lsmedia -lvolmgt -ladm -lefi

26 CERRWARN += -_gcc=-Wno-uninitialized

40 LINTFLAGS += -u
41 CPPFLAGS += -D_FILE_OFFSET_BITS=64

43 $(ROOTBIN)/rmformat := FILEMODE = 04555

45 .KEEP_STATE:

47 all: $(PROG)

49 $(PROG): $(OBJS)
50     $(LINK.c) -o $(PROG) $(OBJS) $(LDLIBS)
51     $(POST_PROCESS)

53 install: all $(ROOTPROG)

55 clean:
56     $(RM) $(OBJS)
```

new/usr/src/cmd/rmformat/Makefile

2

```
58 lint: lint_SRCS

60 $(POFILE) : $(SRCS)
61     $(RM) $@
62     $(COMPILE.cpp) $(SRCS) | $(XGETTEXT) $(XGETFLAGS) -
63     $(SED) -e '/^domain/d' messages.po > $@
64     $(RM) messages.po

66 sb: $(SRCS)
67     $(COMPILE.c) -xsbfast $(SRCS)

69 include ../Makefile.targ
```



```

*****
30262 Tue Apr 22 07:25:48 2014
new/usr/src/cmd/rmformat/rmf_menu.c
4833 Remove volrmmount
Reviewed by: Dan McDonald <dammcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
*****
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 2009 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 *
25 * Copyright 2014 Andrew Stormont.
26 #endif /* ! codereview */
27 */
28
29 /*
30  * rmf_menu.c :
31  * Command line options to rmformat are processed in this file.
32  */
33
34 #include "rmformat.h"
35 #include <sys/smedia.h>
36 #include <priv_utils.h>
37
38 extern int32_t D_flag;
39 extern int32_t e_flag;
40 extern int32_t H_flag;
41 extern int32_t U_flag;
42 extern int32_t V_flag;
43 extern int32_t b_flag;
44 extern int32_t w_flag;
45 extern int32_t W_flag;
46 extern int32_t s_flag;
47 extern int32_t c_flag;
48 extern int32_t F_flag;
49 extern int32_t R_flag;
50 extern int32_t p_flag;
51 extern int32_t l_flag;
52
53 extern char *myname;
54 extern char *slice_file;
55 extern diskaddr_t repair_blk_no;
56 extern int32_t quick_format;
57 extern int32_t long_format;

```

```

58 extern int32_t force_format;
59 extern int32_t rw_protect_enable;
60 extern int32_t rw_protect_disable;
61 extern int32_t wp_enable_passwd;
62 extern int32_t wp_disable_passwd;
63 extern int32_t wp_enable;
64 extern int32_t wp_disable;
65 extern int32_t verify_write;
66 extern char *dev_name;
67 extern char *label;
68 extern int total_devices_found;
69 extern int removable_found;
70 char *global_intr_msg;
71 smmedium_prop_t med_info;
72 int vol_running;
73
74 extern void check_invalid_combinations();
75 extern void check_invalid_combinations_again(int32_t);
76 extern void process_options();
77 extern void get_passwd(struct smwp_state *wp, int32_t confirm);
78 extern int32_t valid_slice_file(smedia_handle_t, int32_t, char *,
79     struct extvtoc *);
80 extern void trap_SIGINT();
81 extern void release_SIGINT();
82 extern int32_t verify(smedia_handle_t handle, int32_t fd,
83     diskaddr_t start_sector, uint32_t nblocks,
84     char *buf, int32_t flag, int32_t blocksize, int32_t no_raw_rw);
85 extern void my_perror(char *err_string);
86 extern void write_default_label(smedia_handle_t, int32_t fd);
87 extern int find_device(int defer, char *tmpstr);
88
89 void overwrite_metadata(int32_t fd, smedia_handle_t handle);
90
91 int32_t write_sunos_label(int32_t fd, int32_t media_type);
92
93 int32_t my_open(char *device_name, int32_t flags);
94 int32_t check_and_unmount_vold(char *device_name, int32_t flag);
95 int32_t check_and_unmount_scsi(char *device_name, int32_t flag);
96
97 int32_t check_and_unmount_floppy(int32_t fd, int32_t flag);
98 int32_t get_confirmation(void);
99
100
101 static void process_F_flag(smedia_handle_t handle, int32_t fd);
102 static void process_w_flag(smedia_handle_t handle);
103 static void process_W_flag(smedia_handle_t handle);
104 static void process_R_flag(smedia_handle_t handle);
105 void process_p_flag(smedia_handle_t handle, int32_t fd);
106 static void process_c_flag(smedia_handle_t handle);
107 static void process_V_flag(smedia_handle_t handle, int32_t fd);
108 static void process_s_flag(smedia_handle_t, int32_t fd);
109 static void process_e_flag(smedia_handle_t handle);
110 static void process_H_flag(smedia_handle_t handle, int32_t fd);
111 static void process_D_flag(smedia_handle_t handle, int32_t fd);
112 static void process_b_flag(int32_t fd);
113 static void process_l_flag(void);
114
115 void
116 process_options()
117 {
118     int32_t fd;
119     smedia_handle_t handle;
120     int32_t m_scsi_umount = 0;
121     int32_t m_flp_umount = 0;
122     int32_t v_device_umount = 0;
123     int32_t umount_required = 0;

```

```

124     int32_t removable;
125     int32_t umount_failed = 0;
126     struct dk_minfo media;

128     check_invalid_combinations();

130     if (l_flag && !dev_name) {
131         process_l_flag();
132         return;
133     }

135     if (U_flag) {
136         if (!(F_flag || H_flag || D_flag)) {
137             F_flag = 1;
138             long_format = 1;
139         }
140     }

142     if (F_flag || w_flag || W_flag || R_flag || D_flag || H_flag ||
143         V_flag || c_flag || b_flag || s_flag || e_flag) {
144         umount_required = 1;
145     }

147     fd = my_open(dev_name, O_RDONLY|O_NDELAY);
148     if (fd < 0) {
149         PERROR("Could not open device");
150         (void) close(fd);
151         exit(1);
152     }

154     if (ioctl(fd, DKIOCREMOVABLE, &removable) < 0) {
155         PERROR("DKIOCREMOVABLE ioctl failed");
156         (void) close(fd);
157         exit(1);
158     }
159     if (!removable) {
160         (void) fprintf(stderr,
161             gettext("Not a removable media device\n"));
162         (void) close(fd);
163         exit(1);
164     }

166     if (ioctl(fd, DKIOCGMEDIAINFO, &media) < 0) {
167         (void) fprintf(stderr,
168             gettext("No media in specified device\n"));
169         (void) close(fd);
170         exit(1);
171     }

173     /* Check if volume manager has mounted this */
174     if (umount_required) {
175         v_device_umount = check_and_unmount_void(dev_name, U_flag);
176         if (v_device_umount != 1) {
177             m_scsi_umount = check_and_unmount_scsi(dev_name,
178                 U_flag);
179             if (m_scsi_umount != 1) {
180                 m_flp_umount = check_and_unmount_floppy(fd,
181                     U_flag);
182                 if (m_flp_umount != 1) {
183                     umount_failed = 1;
184                 }
185             }
186         }
187     }

189     if (umount_required && U_flag && umount_failed) {

```

```

190         if (v_device_umount || m_scsi_umount || m_flp_umount) {
191             (void) fprintf(stderr,
192                 gettext("Could not unmount device.\n"));
193             (void) close(fd);
194             exit(1);
195         }
196     }

198     if (umount_required && !U_flag) {
199         if (v_device_umount || m_scsi_umount || m_flp_umount) {
200             (void) fprintf(stderr, gettext("Device mounted.\n"));
201             (void) fprintf(stderr,
202                 gettext("Requested operation can not be \
203 performed on a mounted device.\n"));
204             (void) close(fd);
205             exit(1);
206         }
207     }
208     /* register the fd with the libsmmedia */
209     handle = smedia_get_handle(fd);
210     if (handle == NULL) {
211         (void) fprintf(stderr,
212             gettext("Failed to get libsmmedia handle.\n"));
213         (void) close(fd);
214         exit(1);
215     }

217     if (smedia_get_medium_property(handle, &med_info) < 0) {
218         (void) fprintf(stderr,
219             gettext("Get medium property failed\n"));
220         (void) smedia_release_handle(handle);
221         (void) close(fd);
222         exit(1);
223     }

225     DPRINTF1("media type %x\n", med_info.sm_media_type);
226     DPRINTF1("media block size %x\n", med_info.sm_blocksize);
227     DPRINTF1("media capacity %u\n", (uint32_t)med_info.sm_capacity);
228     DPRINTF3("media cyl %d head %d sect %d\n",
229         med_info.sm_p cyl, med_info.sm_nhead, med_info.sm_nsect);
230     check_invalid_combinations_again(med_info.sm_media_type);

232     /*
233      * Special handling for pcmcia, sometimes open the file in
234      * read-write mode.
235      */

237     if (med_info.sm_media_type == SM_PCMCIA_MEM) {
238         if (F_flag || H_flag || D_flag || (V_flag && verify_write)) {
239             (void) close(fd);
240             DPRINTF("Reopening device\n");
241             fd = my_open(dev_name, O_RDWR|O_NDELAY);
242             if (fd < 0) {
243                 PERROR("Could not open device");
244                 (void) smedia_release_handle(handle);
245                 (void) close(fd);
246                 exit(1);
247             }
248         }
249     }

251     if (med_info.sm_media_type == SM_PCMCIA_ATA) {
252         if (V_flag || c_flag) {
253             (void) fprintf(stderr,
254                 gettext("Option not supported on PC ATA cards\n"));
255             (void) smedia_release_handle(handle);

```

```

256         (void) close(fd);
257         exit(1);
258     }
259     if (F_flag) {
260         /* same text as used by the format command */
261         (void) fprintf(stderr,
262             gettext("Cannot format this drive. Please use your \
263 Manufacturer supplied formatting utility.\n"));
264         (void) smedia_release_handle(handle);
265         (void) close(fd);
266         exit(1);
267     }
268 }

270 if (F_flag)
271     process_F_flag(handle, fd);
272 if (w_flag)
273     process_w_flag(handle);
274 if (W_flag)
275     process_W_flag(handle);
276 if (R_flag)
277     process_R_flag(handle);
278 if (p_flag)
279     process_p_flag(handle, fd);
280 if (D_flag)
281     process_D_flag(handle, fd);
282 if (H_flag)
283     process_H_flag(handle, fd);
284 if (V_flag)
285     process_V_flag(handle, fd);
286 if (c_flag)
287     process_c_flag(handle);
288 if (b_flag)
289     process_b_flag(fd);
290 if (s_flag)
291     process_s_flag(handle, fd);
292 if (e_flag)
293     process_e_flag(handle);
294 if (l_flag) {
295     process_l_flag();
296 }

298 (void) smedia_release_handle(handle);
299 (void) close(fd);
300 }

302 /*
303 * This routine handles the F_flag.
304 * This options should not be used for floppy. However,
305 * if this option is used for floppy, the option will
306 * be forced to SM_FORMAT_HD and smedia_format is called.
307 * Note that smedia_format is a blocked mode format and it
308 * returns only after the complete formatting is over.
309 */

311 static void
312 process_F_flag(smedia_handle_t handle, int32_t fd)
313 {
314     uint32_t format_flag = 0;
315     uint32_t format_flag;
316     int32_t old_per = 0;
317     int32_t new_per, ret_val;

318     if (force_format) {
319         (void) fprintf(stderr,
320             gettext("Formatting disk.\n"));

```

```

321     } else {
322         (void) fprintf(stderr,
323             gettext("Formatting will erase all the data on disk.\n"));
324         if (!get_confirmation())
325             return;
326     }

328     if (quick_format)
329         format_flag = SM_FORMAT_QUICK;
330     else if (long_format)
331         format_flag = SM_FORMAT_LONG;
332     else if (force_format)
333         format_flag = SM_FORMAT_FORCE;

335     if (med_info.sm_media_type == SM_FLOPPY)
336         format_flag = SM_FORMAT_HD;

338     if ((med_info.sm_media_type != SM_FLOPPY) &&
339         (med_info.sm_media_type != SM_PCMLCIA_MEM) &&
340         (med_info.sm_media_type != SM SCSI_FLOPPY)) {
341         global_intr_msg = "Interrupting format may render the \
342 medium useless";
343     } else {
344         global_intr_msg = "";
345     }
346     trap_SIGINT();

348     if (smedia_format(handle, format_flag, SM_FORMAT_IMMEDIATE) != 0) {
349         if (errno == EINVAL) {
350             (void) fprintf(stderr, gettext("Format failed.\n"));
351             (void) fprintf(stderr, gettext("The medium may not \
352 be compatible for format operation.\n"));
353             (void) fprintf(stderr, gettext("read/write surface \
354 scan may be used to get the effect of formatting.\n"));
355         } else {
356             PERROR("Format failed");
357         }
358         (void) smedia_release_handle(handle);
359         (void) close(fd);
360         exit(1);
361     }

363     /* CONSTCOND */
364     while (1) {
365         ret_val = smedia_check_format_status(handle);
366         if (ret_val == -1) {
367             if (errno != ENOTSUP) {
368                 PERROR("Format failed");
369                 (void) smedia_release_handle(handle);
370                 (void) close(fd);
371                 exit(1);
372             } else {
373                 /* Background formatting is not supported */
374                 break;
375             }
376         }
377         if (ret_val == 100) {
378             (void) printf("\n");
379             (void) fflush(stdout);
380             break;
381         }
382         new_per = (ret_val * 80)/100;
383         while (new_per >= old_per) {
384             (void) printf(".");
385             (void) fflush(stdout);
386             old_per++;

```

```

387     }
388     (void) sleep(6);
389 }

391 if ((med_info.sm_media_type == SM_FLOPPY) ||
392     (med_info.sm_media_type == SM_PCPCIA_MEM) ||
393     (med_info.sm_media_type == SM SCSI_FLOPPY)) {
394     (void) write_sunos_label(fd, med_info.sm_media_type);
395 } else {

397     /*
398     * Iomega drives don't destroy the data in quick format.
399     * Do a best effort write to first 1024 sectors.
400     */

402     if (quick_format)
403         overwrite_metadata(fd, handle);

405     (void) write_default_label(handle, fd);
406 }

408 release_SIGINT();
409 }

411 /*
412 * List removable devices.
413 */
414 static void
415 process_l_flag()
416 {
417     int retry;
418     int removable;
419     int total_devices_found_last_time;
420     int defer = 0;
421     char *tmpstr = NULL;
422     char *tmpstr;

423 #define MAX_RETRIES_FOR_SCANNING 3

425     vol_running = volmgt_running();
426     if (vol_running)
427         defer = 1;
428     (void) printf(gettext("Looking for devices...\n"));
429     total_devices_found_last_time = 0;

431     /*
432     * Strip out any leading path. For example, /dev/rdisk/c3t0d0s2
433     * will result in tmpstr = c3t0d0s2. dev_name is given as input
434     * argument.
435     */
436     if (dev_name) {
437         if ((tmpstr = strrchr(dev_name, '/')) != NULL) {
438             tmpstr += sizeof(char);
439         } else {
440             tmpstr = dev_name;
441         }
442     }

444     for (retry = 0; retry < MAX_RETRIES_FOR_SCANNING; retry++) {
445         removable = find_device(defer, tmpstr);
446         if (removable == -1)
447             break;

449     /*
450     * We'll do a small sleep and retry the command if volume
451     * manager is running and no removable devices are found.

```

```

452     * This is because the device may be busy.
453     */
454     if (defer || (vol_running && (removable == 0))) {
455         if ((total_devices_found == 0) ||
456             (total_devices_found !=
457              total_devices_found_last_time)) {
458             total_devices_found_last_time =
459                 total_devices_found;
460             (void) sleep(2);
461         } else {
462             /* Do the printing this time */
463             defer = 0;
464             removable_found = 0;
465         }

467     } else
468         break;
469 }
470 if (removable_found == 0)
471     (void) printf(gettext("No removables found.\n"));
472 }

```

unchanged portion omitted

```

*****
46713 Tue Apr 22 07:25:48 2014
new/usr/src/cmd/rmformat/rmf_misc.c
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 *
21 *
22 */
23
24 /*
25 * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
26 * Use is subject to license terms.
27 *
28 *
29 */
30 * rmf_misc.c :
31 * Miscellaneous routines for rmformat.
32 */
33
34 #include <sys/types.h>
35 #include <stdio.h>
36 #include <sys/mnttab.h>
37 #include <volmgt.h>
38 #include <sys/dkio.h>
39 #include <sys/fdio.h>
40 #include <sys/vtoc.h>
41 #include <sys/termios.h>
42 #include <sys/mount.h>
43 #include <ctype.h>
44 #include <signal.h>
45 #include <sys/wait.h>
46 #include <dirent.h>
47 #include <priv_utils.h>
48 #include <stdarg.h>
49 #include "rmformat.h"
50
51 /*
52 * Definitions.
53 */
54 #define SENSE_KEY(rqbuf)      (rqbuf[2] & 0xf) /* scsi error category */
55 #define ASC(rqbuf)           (rqbuf[12]) /* additional sense code */
56 #define ASCQ(rqbuf)          (rqbuf[13]) /* ASC qualifier */

```

```

58 #define DEFAULT_SCSI_TIMEOUT 60
59 #define INQUIRY_CMD          0x12
60 #define RQBUFLEN            32
61 #define CD_RW                1 /* CD_RW/CD-R */
62 #define WRITE_10_CMD        0x2A
63 #define READ_INFO_CMD       0x51
64 #define SYNC_CACHE_CMD      0x35
65 #define CLOSE_TRACK_CMD     0x5B
66 #define MODE_SENSE_10_CMD   0x5A
67 #define DEVFS_PREFIX        "/devices"
68
69 int          uscsi_error; /* used for debugging failed uscsi */
70 char         rqbuf[RQBUFLEN];
71 static uint_t total_retries;
72 static struct uscsi_cmd uscmd;
73 static char   ucdb[16];
74 uchar_t      uscsi_status, rqstatus, rgresid;
75 int          total_devices_found = 0;
76 int          removable_found = 0;
77
78 extern char   *global_intr_msg;
79 extern int     vol_running;
80 extern char    *dev_name;
81 extern int32_t m_flag;
82
83 /*
84 * ON-private functions from libvolmgt
85 */
86 int      _dev_mounted(char *path);
87
88 /*
89 * Function prototypes.
90 */
91 static int my_umount(char *mountp);
92 static int my_rmmount(char *real_name);
93 static int my_volrmmount(char *real_name);
94 static int vol_name_to_dev_node(char *vname, char *found);
95 static int vol_lookup(char *supplied, char *found);
96 static int *get_device(char *user_supplied, char *node);
97 static char *get_physical_name(char *path);
98 static int lookup_device(char *supplied, char *found);
99 static void fini_device(device_t *dev);
100 static int is_cd(char *node);
101 void *my_zalloc(size_t size);
102 void err_msg(char *fmt, ...);
103 int inquiry(int fd, uchar_t *inq);
104 struct uscsi_cmd *get_uscsi_cmd(void);
105 int uscsi(int fd, struct uscsi_cmd *scmd);
106 int get_mode_page(int fd, int page_no, int pc, int buf_len,
107 uchar_t *buffer);
108 int mode_sense(int fd, uchar_t pc, int dbd, int page_len,
109 uchar_t *buffer);
110 uint16_t read_scsil6(void *addr);
111 int check_device(device_t *dev, int cond);
112 static void get_media_info(device_t *t_dev, char *sdev,
113 char *pname, char *sn);
114
115 extern void process_p_flag(smedia_handle_t handle, int32_t fd);
116
117 void my_perror(char *err_string)
118 {
119
120     int error_no;
121     if (errno == 0)

```

```

122         return;

124     error_no = errno;
125     (void) fprintf(stderr, "%s", err_string);
126     (void) fprintf(stderr, gettext(" : "));
127     errno = error_no;
128     perror("");
129 }
    unchanged_portion_omitted_

228 int32_t
229 check_and_unmount_vold(char *device_name, int32_t flag)
230 {
231     char *real_name;
232     char *nm;
233     char tmp_path_name[PATH_MAX];
234     struct stat stat_buf;
235     int32_t ret_val = 0;
236     struct mnttab *mntp;
237     FILE *fp;
238     int nl;

240     DPRINTF1("Device name %s\n", device_name);

242     if (volmgt_running() == 0) {
243         DPRINTF("Vold not running\n");
244         return (0);
245     }
246     if ((nm = volmgt_symname(device_name)) == NULL) {
247         DPRINTF("path not managed\n");
248         real_name = media_findname(device_name);
249     } else {
250         DPRINTF1("path managed as %s\n", nm);
251         real_name = media_findname(nm);
252         DPRINTF1("real name %s\n", real_name);
253     }

255     if (real_name == NULL)
256         return (-1);

258     /*
259     * To find out whether the device has been mounted by
260     * volume manager...
261     *
262     * Convert the real name to a block device address.
263     * Do a partial match with the mnttab entries.
264     * Make sure the match is in the beginning to avoid if
265     * anybody puts a label similiar to volume manager path names.
266     * Then use "rmmount -u <dev_name>" if -U flag is set.
267     * Then use "volrmmount -e <dev_name>" if -U flag is set.
268     */

269     nl = strlen("/vol/dev/");

271     if (strncmp(real_name, "/vol/dev/", nl) != 0)
272         return (0);
273     if (real_name[nl] == 'r') {
274         (void) snprintf(tmp_path_name, PATH_MAX, "%s%s", "/vol/dev/",
275             &real_name[nl + 1]);
276     } else {
277         (void) snprintf(tmp_path_name, PATH_MAX, "%s", real_name);
278     }
279     DPRINTF1("%s\n", tmp_path_name);
280     ret_val = stat(tmp_path_name, &stat_buf);
281     if (ret_val < 0) {
282         PERROR("Could not stat");

```

```

283         return (-1);
284     }

286     fp = fopen("/etc/mnttab", "r");

288     if (fp == NULL) {
289         PERROR("Could not open /etc/mnttab");
290         return (-1);
291     }

293     mntp = (struct mnttab *)malloc(sizeof (struct mnttab));
294     if (mntp == NULL) {
295         PERROR("malloc failed");
296         (void) fclose(fp);
297         return (-1);
298     }
299     errno = 0;
300     while (getmntent(fp, mntp) == 0) {
301         if (errno != 0) {
302             PERROR("Error with mnttab");
303             (void) fclose(fp);
304             return (-1);
305         }
306         /* Is it a probable entry? */
307         DPRINTF1(" %s\n", mntp->mnt_special);
308         if (strstr(mntp->mnt_special, tmp_path_name) !=
309             mntp->mnt_special) {
310             /* Skip to next entry */
311             continue;
312         } else {
313             DPRINTF1("Found!! %s\n", mntp->mnt_special);
314             ret_val = 1;
315             break;
316         }
317     }

319     if (ret_val == 1) {
320         if (flag) {
321             if (my_rmmount(real_name) < 0) {
322                 if (my_volrmmount(real_name) < 0) {
323                     ret_val = -1;
324                 }
325             } else {
326                 ret_val = -1;
327             }
328             (void) fclose(fp);
329             free(mntp);
330             return (ret_val);
331         }
    unchanged_portion_omitted_

1072 static int
1073 my_rmmount(char *real_name)
1074 my_volrmmount(char *real_name)
1075 {
1076     int pid, rval;

1077     /* Turn on the privileges. */
1078     (void) __priv_bracket(PRIV_ON);

1080     pid = fork();

1082     /* Turn off the privileges. */
1083     (void) __priv_bracket(PRIV_OFF);

```

```

1085     /* create a child to unmount the path */
1086     if (pid < 0) {
1087         PERROR("fork failed");
1088         exit(0);
1089     }
1091     if (pid == 0) {
1092         /* the child */
1093         /* get rid of those nasty err messages */
1094         DPRINTF1("call_unmount_prog: calling %s\n",
1095             "/usr/bin/rmmount");
1096         DPRINTF1("call_unmount_prog: calling %s \n",
1097             "/usr/bin/volrmmount");
1098         /* Turn on the privileges. */
1099         (void) __priv_bracket(PRIV_ON);
1100         if (execl("/usr/bin/rmmount", "/usr/bin/rmmount", "-u",
1101             if (execl("/usr/bin/volrmmount", "/usr/bin/volrmmount", "-e",
1102                 real_name, NULL) < 0) {
1103             PERROR("rmmount exec failed");
1104             PERROR("volrmmount exec failed");
1105             /* Turn off the privileges */
1106             (void) __priv_bracket(PRIV_OFF);
1107             exit(-1);
1108         } else if (waitpid(pid, &rval, 0) == pid) {
1109             if (WIFEXITED(rval)) {
1110                 if (WEXITSTATUS(rval) == 0) {
1111                     DPRINTF("rmmount: Success\n");
1112                     DPRINTF("volrmmount: Success\n");
1113                     return (1);
1114                 }
1115             }
1116         }
1117     }
1118     return (-1);
1119 }
1120
1121 int
1122 find_device(int defer, char *tmpstr)
1123 {
1124     DIR *dir;
1125     struct dirent *dirent;
1126     char sdev[PATH_MAX], dev[PATH_MAX], *pname;
1127     device_t *t_dev;
1128     int removable = 0;
1129     int device_type = 0;
1130     int hotpluggable = 0;
1131     struct dk_minfo mediainfo;
1132     static int found = 0;
1133
1134     dir = opendir("/dev/rdsk");
1135     if (dir == NULL)
1136         return (-1);
1137
1138     total_devices_found = 0;
1139     while ((dirent = readdir(dir)) != NULL) {
1140         if (dirent->d_name[0] == '.') {
1141             continue;
1142         }
1143         (void) snprintf(sdev, PATH_MAX, "/dev/rdsk/%s",
1144             dirent->d_name);
1145 #ifdef sparc
1146         if (!strstr(sdev, "s2")) {
1147             continue;
1148         }
1149 #else /* x86 */

```

```

1146         if (vol_running) {
1147             if (!(strstr(sdev, "s2") || strstr(sdev, "p0"))) {
1148                 continue;
1149             }
1150         } else {
1151             if (!strstr(sdev, "p0")) {
1152                 continue;
1153             }
1154         }
1155 #endif
1156         if (!lookup_device(sdev, dev)) {
1157             continue;
1158         }
1159         if ((t_dev = get_device(NULL, dev)) == NULL) {
1160             continue;
1161         }
1162         total_devices_found++;
1163
1164         if ((!defer) && !found) {
1165             char *sn, *tmpbuf = NULL;
1166             char *sn, *tmpbuf;
1167             /*
1168              * dev_name is an optional command line input.
1169              */
1170             if (dev_name) {
1171                 if (strstr(dirent->d_name, tmpstr)) {
1172                     found = 1;
1173                 } else if (!vol_running) {
1174                     continue;
1175                 }
1176             }
1177             /*
1178              * volmgt_symname() returns NULL if the device
1179              * is not managed by volmgt.
1180              */
1181             sn = volmgt_symname(sdev);
1182
1183             if (vol_running && (sn != NULL)) {
1184                 if (strstr(sn, "dev") == NULL) {
1185                     tmpbuf = (char *)my_zalloc(PATH_MAX);
1186                     (void) strcpy(tmpbuf,
1187                         "/vol/dev/aliases/");
1188                     (void) strcat(tmpbuf, sn);
1189                     free(sn);
1190                     sn = tmpbuf;
1191                 }
1192                 if (dev_name && !found) {
1193                     if (!strstr(tmpbuf, tmpstr)) {
1194                         continue;
1195                     } else {
1196                         found = 1;
1197                     }
1198                 }
1199             }
1200         }
1201         /*
1202          * Get device type information for CD/DVD devices.
1203          */
1204         if (is_cd(dev)) {
1205             if (check_device(t_dev,
1206                 CHECK_DEVICE_IS_DVD_WRITABLE)) {
1207                 device_type = DK_DVDR;
1208             } else if (check_device(t_dev,
1209                 CHECK_DEVICE_IS_DVD_READABLE)) {
1210                 device_type = DK_DVDROM;
1211             } else if (check_device(t_dev,

```

```

1211         CHECK_DEVICE_IS_CD_WRITABLE)) {
1212             device_type = DK_CDR;
1213         } else {
1214             device_type = DK_CDROM;
1215         }
1216     } else {
1217         device_type = ioctl(t_dev->d_fd,
1218             DKIOCGMEDIAINFO, &mediainfo);
1219         if (device_type < 0)
1220             device_type = 0;
1221         else
1222             device_type = mediainfo.dki_media_type;
1223     }
1224
1225     if (!ioctl(t_dev->d_fd, DKIOCREMOVABLE, &removable) &&
1226         !ioctl(t_dev->d_fd, DKIOCHOTPLUGGABLE,
1227             &hotpluggable)) {
1228         if (removable || hotpluggable) {
1229             removable_found++;
1230             pname = get_physical_name(sdev);
1231             if (sn) {
1232                 (void) printf(" %4d. "
1233                     "Volmgt Node: %s\n",
1234                     removable_found, sn);
1235                 (void) printf(" "
1236                     "Logical Node: %s\n", sdev);
1237                 (void) printf(" "
1238                     "Physical Node: %s\n",
1239                     pname);
1240             } else {
1241                 (void) printf(" %4d. "
1242                     "Logical Node: %s\n",
1243                     removable_found, sdev);
1244                 (void) printf(" "
1245                     "Physical Node: %s\n",
1246                     pname);
1247             }
1248             (void) printf("          Connected "
1249                 "Device: %-8.8s %-16.16s "
1250                 "%-4.4s\n",
1251                 &t_dev->d_inq[8],
1252                 &t_dev->d_inq[16],
1253                 &t_dev->d_inq[32]);
1254             (void) printf("          Device "
1255                 "Type: ");
1256         } else
1257             continue;
1258     } else
1259         continue;
1260
1261     switch (device_type) {
1262     case DK_CDROM:
1263         (void) printf("CD Reader\n");
1264         break;
1265     case DK_CDR:
1266     case DK_CDRW:
1267         (void) printf("CD Reader/Writer\n");
1268         break;
1269     case DK_DVDROM:
1270         (void) printf("DVD Reader\n");
1271         break;
1272     case DK_DVDR:
1273     case DK_DVDRAM:
1274         (void) printf("DVD Reader/Writer\n");
1275         break;
1276     case DK_FIXED_DISK:

```

```

1277         if (strstr((const char *)
1278             &t_dev->d_inq[16], "FD") ||
1279             strstr((const char *)
1280             &t_dev->d_inq[16], "LS-120"))
1281             (void) printf("Floppy "
1282                 "drive\n");
1283         else
1284             (void) printf("Removable\n");
1285         break;
1286     case DK_FLOPPY:
1287         (void) printf("Floppy drive\n");
1288         break;
1289     case DK_ZIP:
1290         (void) printf("Zip drive\n");
1291         break;
1292     case DK_JAZ:
1293         (void) printf("Jaz drive\n");
1294         break;
1295     default:
1296         (void) printf("<Unknown>\n");
1297         DPRINTF1("\t %d\n", device_type);
1298         break;
1299     }
1300     get_media_info(t_dev, sdev, pname, sn);
1301     }
1302     fini_device(t_dev);
1303 }
1304
1305     (void) closedir(dir);
1306     return (removable_found);
1307 }

```

unchanged portion omitted


```

*****
39066 Tue Apr 22 07:25:48 2014
new/usr/src/cmd/rmformat/rmf_slice.c
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
*****
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 2009 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 *
25 * Copyright 2014 Andrew Stormont.
26 #endif /* ! codereview */
27 */
28
29 /*
30  * rmf_slice.c :
31  * This file contains the functions for parsing a slice file
32  * for rmformat.
33  */
34
35 #include <sys/types.h>
36 #include <ctype.h>
37 #include <sys/vtoc.h>
38 #include <stdlib.h>
39 #include <unistd.h>
40 #include <string.h>
41 #include <fcntl.h>
42 #include <errno.h>
43 #include <memory.h>
44 #include <dirent.h>
45 #include <sys/fcntl.h>
46 #include <sys/param.h>
47 #include <sys/stat.h>
48 #include <stdio.h>
49 #include <sys/dkio.h>
50 #include <priv_utils.h>
51 #include "rmformat.h"
52
53 extern void my_perror(char *err_string);
54
55 static int32_t last_token_type = 0;
56 #define spc() (last_token_type)

```

```

59 /*
60  * This global is used to store the current line # in the
61  * data file. It must be global because the I/O routines
62  * are allowed to side effect it to keep track of backslashed
63  * newlines.
64  */
65
66 static int32_t data_lineno; /* current line # in data file */
67
68 #define CHG_MODE_UNDEFINED (-1) /* undefined value */
69 #define CHG_MODE_SET 0 /* set bits by or'ing */
70 #define CHG_MODE_CLR 1 /* clr bits by and'ing */
71 #define CHG_MODE_ABS 2 /* set absolute value */
72
73
74 #define TOKEN_SIZE 36 /* max length of a token */
75 typedef char TOKEN[TOKEN_SIZE+1]; /* token type */
76 #define DATA_INPUT 0 /* 2 modes of input */
77 #define CMD_INPUT 1
78 #define WILD_STRING "$" /* wildcard character */
79 #define COMMENT_CHAR '#' /* comment character */
80
81 /*
82  * List of strings with arbitrary matching values
83  */
84 typedef struct slist {
85     char *str;
86     char *help;
87     int32_t value;
88 } slist_t;
89
90 static slist_t ptag_choices[] = {
91     {"unassigned", "", V_UNASSIGNED },
92     {"boot", "", V_BOOT },
93     {"root", "", V_ROOT },
94     {"swap", "", V_SWAP },
95     {"usr", "", V_USR },
96     {"backup", "", V_BACKUP },
97     {"stand", "", V_STAND },
98     {"var", "", V_VAR },
99     {"home", "", V_HOME },
100    {"alternates", "", V_ALTSCTR },
101    {NULL }
102 };
103
104
105 /*
106  * Choices for the p_flag vtoc field
107  */
108 static slist_t pflag_choices[] = {
109     {"wm", "read-write, mountable", 0 },
110     {"wu", "read-write, unmountable", V_UNMNT },
111     {"rm", "read-only, mountable", V_RDONLY },
112     {"ru", "read-only, unmountable", V_RDONLY|V_UNMNT },
113     {NULL }
114 };
115
116 /*
117  * The definitions are the token types that the data file parser recognizes.
118  */
119 #define SUP_EOF -1 /* eof token */
120 #define SUP_STRING 0 /* string token */
121 #define SUP_EQ 1 /* equals token */
122 #define SUP_COMMA 2 /* comma token */
123 #define SUP_COLON 3 /* colon token */

```

```

124 #define SUP_EOL          4          /* newline token */
125 #define SUP_OR           5          /* vertical bar */
126 #define SUP_AND          6          /* ampersand */
127 #define SUP_TILDE       7          /* tilde */

130 /*
131  * Prototypes for ANSI C compilers
132  */
133 static int32_t sup_prxfile(char *file_name, struct extvtoc *vt);
134 static int32_t sup_setpart(struct extvtoc *vt);
135 static void sup_pushchar(int32_t c);
136 static void clean_token(char *cleantoken, char *token);
137 static void clean_token(char *cleantoken, char *token);
138 static int32_t sup_inputchar();
139 static int32_t sup_gettoken(char *buf);
140 static int32_t sup_get_token(char *buf);
141 static int32_t find_value(slist_t *slist, char *str, int32_t *value);
142 static int32_t check_vtoc_sanity(smedia_handle_t, int32_t fd,
143     struct extvtoc *vt);
144 static uint64_t str2sector(char *str);
145 static int32_t strcnt(char *s1, char *s2);
146 static int32_t get_fdisk(smedia_handle_t, int32_t fd, int32_t offset,
147     struct fdisk_info *fdisk);
148 static void erase(smedia_handle_t handle, diskaddr_t offset, diskaddr_t size);

150 extern char *myname;
151 extern uint64_t my_atoll(char *ptr);
152 extern smmedium_prop_t med_info;

154 static FILE *data_file;

156 static int32_t
157 sup_prxfile(char *file_name, struct extvtoc *vt)
158 {
159     int32_t status, ret_val;
160     TOKEN token;
161     TOKEN cleaned;

163     /*
164      * Open the data file. Return 0 if unable to do so.
165      */
166     data_file = fopen(file_name, "r");
167     if (data_file == NULL) {
168         PERROR("Open failed");
169         return (-1);
170     }
171     /*
172      * Step through the data file a meta-line at a time. There are
173      * typically several backslashed newlines in each meta-line,
174      * so data_lineno will be getting side effected along the way.
175      */
176     data_lineno = 1;
177     for (;;) {

179         /*
180          * Get the keyword.
181          */
182         status = sup_gettoken(token);
183         /*
184          * If we hit the end of the data file, we're done.
185          */
186         if (status == SUP_EOF)
187             break;
188         /*
189          * If the line starts with some key character, it's an error.

```

```

190         /*
191          * if (status != SUP_STRING) {
192             (void) fprintf(stderr,
193                 gettext("Expecting keyword, found '%s'"),
194                 token);
195             (void) fprintf(stderr,
196                 gettext("Line no %d\n"), data_lineno);
197             continue;
198         }
199         /*
200          * Clean up the token and see which keyword it is. Call
201          * the appropriate routine to process the rest of the line.
202          */
203         clean_token(cleaned, token);
204         if (strcmp(cleaned, "slices") == 0) {
205             ret_val = sup_setpart(vt);
206             (void) fclose(data_file);
207             return (ret_val);
208         } else {
209             (void) fprintf(stderr, gettext("Unknown keyword '%s'"),
210                 cleaned);
211             (void) fprintf(stderr,
212                 gettext("Line no %d\n"), data_lineno);
213             (void) fclose(data_file);
214             return (-1);
215         }
216     }
217     /*
218      * Close the data file.
219      */
220     (void) fclose(data_file);

222     (void) fprintf(stderr,
223         gettext("Unexpected end of file (line no %d)\n"), data_lineno);
224     return (-1);
225 }

227 static int32_t
228 sup_gettoken(char *buf)
229 {
230     /*
231      * Skip end of lines and blank lines.
232      */
233     while ((last_token_type = sup_get_token(buf)) == SUP_EOL)
234         ;
235     return (last_token_type);
236 }

238 static int32_t
239 sup_get_token(char *buf)
240 {
241     char *ptr = buf;
242     int32_t c, quoted = 0;

244     /*
245      * Was an end of file detected last try?
246      */

248     if (feof(data_file)) {
249         return (SUP_EOF);
250     }

252     /*
253      * Zero out the returned token buffer
254      */

```

```

256     bzero(buf, TOKEN_SIZE + 1);
257
258     /*
259     * Strip off leading white-space.
260     */
261     while (isspace(c = sup_inputchar()))
262         ;
263
264     /*
265     * Only white spaces and then end of file?
266     */
267
268     if (feof(data_file)) {
269         return (SUP_EOF);
270     }
271
272     /*
273     * Read in characters until we hit unquoted white-space.
274     */
275     for (; !isspace(c) || quoted; c = sup_inputchar()) {
276
277         /*
278         * If we hit eof, check if we have anything in buffer.
279         * if we have, return STRING, next time we will return EOF
280         * else, return EOF here...should not happen.
281         */
282         if (feof(data_file)) {
283             if (ptr - buf > 0) {
284                 return (SUP_STRING);
285             } else {
286                 return (SUP_EOF);
287             }
288         }
289
290         /*
291         * If we hit a double quote, change the state of quoting.
292         */
293         if (c == '"') {
294             quoted = !quoted;
295             continue;
296         }
297         /*
298         * If we hit a newline, that delimits a token.
299         */
300         if (c == '\n')
301             break;
302         /*
303         * If we hit any nonquoted special delimiters, that delimits
304         * a token.
305         */
306         if (!quoted && (c == '=' || c == ',' || c == ':' ||
307             c == '#' || c == '|' || c == '&' || c == '~'))
308             break;
309         /*
310         * Store the character if there's room left.
311         */
312         if (ptr - buf < TOKEN_SIZE)
313             *ptr++ = (char)c;
314     }
315     /*
316     * If we stored characters in the buffer, then we inputted a string.
317     * Push the delimiter back into the pipe and return the string.
318     */
319     if (ptr - buf > 0) {
320         sup_pushchar(c);
321         return (SUP_STRING);

```

```

322     }
323     /*
324     * We didn't input a string, so we must have inputted a known delimiter.
325     * store the delimiter in the buffer, so it will get returned.
326     */
327     buf[0] = c;
328     /*
329     * Switch on the delimiter. Return the appropriate value for each one.
330     */
331     switch (c) {
332     case '=':
333         return (SUP_EQ);
334     case ':':
335         return (SUP_COLON);
336     case ',':
337         return (SUP_COMMA);
338     case '\n':
339         return (SUP_EOL);
340     case '|':
341         return (SUP_OR);
342     case '&':
343         return (SUP_AND);
344     case '~':
345         return (SUP_TILDE);
346     case '#':
347         /*
348         * For comments, we flush out the rest of the line and return
349         * an eol.
350         */
351         while ((c = sup_inputchar()) != '\n' && !feof(data_file))
352             ;
353         if (feof(data_file))
354             return (SUP_EOF);
355         else
356             return (SUP_EOL);
357     /*
358     * Shouldn't ever get here.
359     */
360     default:
361         return (SUP_STRING);
362     }
363 }
364 static int32_t
365 sup_inputchar()
366 {
367     int32_t c;
368
369     /*
370     * Input the character.
371     */
372     c = getc(data_file);
373     /*
374     * If it's not a backslash, return it.
375     */
376
377     /*
378     * It was a backslash. Get the next character.
379     */
380
381     if (c == '\\')
382         c = getc(data_file);
383
384     /*
385     * If it was a newline, update the line counter and get the next
386     * character.
387     */

```

```

388     if (c == '\n') {
389         data_lineno++;
390     }
391     /*
392     * Return the character.
393     */
394     return (c);
395 }

397 static void
398 sup_pushchar(int32_t c)
399 {
401     (void) ungetc(c, data_file);
402     if (c == '\n')
403         data_lineno--;
404 }

406 static void
407 clean_token(char *cleantoken, char *token)
408 {
409     char *ptr;
411     /*
412     * Strip off leading white-space.
413     */
414     for (ptr = token; isspace(*ptr) && (ptr <=
415         (token + strlen(token) - 1)); ptr++)
416         ;
418     /*
419     * Copy it into the clean buffer.
420     */
421     (void) strcpy(cleantoken, ptr);
422     /*
423     * Strip off trailing white-space.
424     */
425     for (ptr = cleantoken + strlen(cleantoken) - 1;
426         isspace(*ptr) && (ptr >= cleantoken); ptr--) {
427         *ptr = '\0';
428     }
429 }

431 static int32_t
432 sup_setpart(struct extvtoc *vt)
433 {
434     TOKEN token, cleaned, ident;
435     int32_t i, index, status;
436     uint64_t val1, val2;
437     ushort_t vtoc_tag = 0xFFFF;
438     ushort_t vtoc_flag = 0xFFFF;
440     /*
441     * Pull in some grammar.
442     */
444     status = sup_gettoken(token);
446     if (status != SUP_COLON) {
447         (void) fprintf(stderr,
448             gettext("Expecting ':', found '%s'", token);
449         (void) fprintf(stderr,
450             gettext("Line no %d\n"), data_lineno);
451         return (-1);
452     }

```

```

454     for (;;) {
455         status = sup_gettoken(token);
456         if (status != SUP_STRING) {
457             (void) fprintf(stderr,
458                 gettext("Expecting string, found '%s'", token);
459             (void) fprintf(stderr,
460                 gettext("Line no %d\n"), data_lineno);
461             return (-1);
462         }
463         clean_token(ident, token);
464         /*
465         * Here's the index of the partition we're dealing with
466         */
467         index = (int32_t)my_atoll(ident);
468         if ((index < 0) || (index >= NDKMAP)) {
469             (void) fprintf(stderr,
470                 gettext("Unknown partition %d", index);
471             (void) fprintf(stderr,
472                 gettext("Line no %d\n"), data_lineno);
473             return (-1);
474         }
475         /*
476         * Check for floppy and PCMCIA_MEM cards.
477         * for floppy, the partition no. can be 0 1 2.
478         * for PCMCIA, the partition no. can be 2
479         */
480         if (med_info.sm_media_type == SM_FLOPPY) {
481             if ((index < 0) || (index > 2)) {
482                 (void) fprintf(stderr, gettext(
483                     "Floppy can have partitions 0 1 and 2\n"));
484                 return (-1);
485             }
486         }
487         if (med_info.sm_media_type == SM_PCPCIA_MEM) {
488             if (index != 2) {
489                 (void) fprintf(stderr, gettext(
490                     "PCMCIA Memory cards can have partition 2 only.\n"));
491                 return (-1);
492             }
493         }
495         DPRINTF1("\n Partition %d: ", index);
497         status = sup_gettoken(token);
498         if (status != SUP_EQUAL) {
499             (void) fprintf(stderr,
500                 gettext("Expecting '=', found '%s'", token);
501             (void) fprintf(stderr,
502                 gettext("Line no %d\n"), data_lineno);
503             return (-1);
505         }
508         status = sup_gettoken(token);
509         /*
510         * If we hit a key character, it's an error.
511         */
512         if (status != SUP_STRING) {
513             (void) fprintf(stderr,
514                 gettext("Expecting value, found '%s'", token);
515             (void) fprintf(stderr,
516                 gettext("Line no %d\n"), data_lineno);
517             return (-1);
518         }
519         clean_token(cleaned, token);

```

```

520      /*
521      * <tag> may be one of: boot, root, swap, etc.
522      * <flag> consists of two characters:
523      *   W (writable) or R (read-only)
524      *   M (mountable) or U (unmountable)
525      *
526      * Start with the defaults assigned above:
527      */
528
529      /*
530      * All other attributes have a pair of numeric values.
531      * Convert the first value to a number. This value
532      * is the starting cylinder number of the partition.
533      */
534
535      /* Check for valid partition, e.g. > 8 or 16 */
536      vall = str2sector(cleaned);
537      if (vall == -1) {
538          (void) fprintf(stderr,
539              gettext("Invalid partition beggining %s \n"),
540              cleaned);
541          (void) fprintf(stderr,
542              gettext("Line no %d\n"), data_lineno);
543      }
544
545      DPRINTF1(" begins %s", cleaned);
546      /*
547      * Pull in some grammar.
548      */
549      status = sup_gettoken(token);
550      if (status != SUP_COMMA) {
551          (void) fprintf(stderr,
552              gettext("Expecting ', ', found '%s'", token);
553          (void) fprintf(stderr,
554              gettext("Line no %d\n"), data_lineno);
555          return (-1);
556      }
557      /*
558      * Pull in the second value.
559      */
560      status = sup_gettoken(token);
561      if (status != SUP_STRING) {
562          (void) fprintf(stderr,
563              gettext("Expecting value, found '%s'", token);
564          (void) fprintf(stderr,
565              gettext("Line no %d\n"), data_lineno);
566          return (-1);
567      }
568      clean_token(cleaned, token);
569
570      val2 = str2sector(cleaned);
571      if (val2 == -1) {
572          (void) fprintf(stderr,
573              gettext("Invalid partition size %s \n"),
574              cleaned);
575          (void) fprintf(stderr,
576              gettext("Line no %d\n"), data_lineno);
577      }
578      DPRINTF1(" ends %s ", cleaned);
579
580      /*
581      * Pull in some grammar.
582      */
583      status = sup_gettoken(token);
584
585      if (status == SUP_COMMA) {

```

```

586      /* tags and flags */
587      status = sup_gettoken(token);
588      if (status != SUP_STRING) {
589          (void) fprintf(stderr,
590              gettext("Expecting value, found '%s'",
591                  token);
592          (void) fprintf(stderr,
593              gettext("Line no %d\n"), data_lineno);
594          return (-1);
595      }
596      clean_token(cleaned, token);
597      if (find_value(pflag_choices, cleaned, &i) == 1) {
598          /*
599          * Found valid tag. Use it and advance parser
600          */
601          DPRINTF1(" flag = %s", cleaned);
602          vtoc_flag = (ushort_t)i;
603          status = sup_gettoken(token);
604      } else if (find_value(ptag_choices, cleaned, &i) == 1) {
605          DPRINTF1(" tag = %s", cleaned);
606          vtoc_tag = (ushort_t)i;
607          status = sup_gettoken(token);
608          if (status == SUP_COMMA) {
609              (void) fprintf(stderr,
610                  gettext("Expecting : got %s\n"),
611                  token);
612              (void) fprintf(stderr,
613                  gettext("Line no %d\n"),
614                  data_lineno);
615              return (-1);
616          }
617      } else {
618          (void) fprintf(stderr,
619              gettext("Invalid flag or tag\n"));
620          (void) fprintf(stderr,
621              gettext("Line no %d\n"), data_lineno);
622          return (-1);
623      }
624
625      if (status == SUP_COMMA) {
626          /* Can be tag only */
627
628          status = sup_gettoken(token);
629          if (status != SUP_STRING) {
630              (void) fprintf(stderr,
631                  gettext("Expecting value"
632                      ", found '%s'",
633                      token);
634              (void) fprintf(stderr,
635                  gettext("Line no %d\n"),
636                  data_lineno);
637              return (-1);
638          }
639
640          clean_token(cleaned, token);
641          if (find_value(ptag_choices,
642              cleaned, &i) == 1) {
643              DPRINTF1(" tag = %s", cleaned);
644              vtoc_tag = (ushort_t)i;
645          }
646      }
647      status = sup_gettoken(token);
648      }
649      }
650
651      /*

```

```

652     * Fill in the appropriate map entry with the values.
653     */
654     vt->v_part[index].p_start = val1;
655     vt->v_part[index].p_size = val2;
656     if (vtoc_tag != 0xFFFF) {
657         vt->v_part[index].p_tag = vtoc_tag;
658         vtoc_tag = 0xFFFF;
659     }
660     if (vtoc_flag != 0xFFFF) {
661         vt->v_part[index].p_flag = vtoc_flag;
662         vtoc_flag = 0xFFFF;
663     }
664     if (status == SUP_EOF) {
665         DPRINTF("\nEnd of file\n");
666         break;
667     }
668     if (status != SUP_COLON) {
669         (void) fprintf(stderr,
670             gettext("Expecting ':', found '%s'", token);
671         (void) fprintf(stderr,
672             gettext("Line no %d\n", data_lineno);
673         return (-1);
674     }
675 }
676 }
677 return (0);
678 }
679
680 static int32_t
681 find_value(slist_t *slist, char *match_str, int32_t *match_value)
682 {
683     int32_t i;
684     int32_t nmatches;
685     int32_t length;
686     int32_t match_length;
687
688     nmatches = 0;
689     length = 0;
690
691     match_length = strlen(match_str);
692
693     for (; slist->str != NULL; slist++) {
694         /*
695          * See how many characters of the token match
696          */
697         i = strncat(match_str, slist->str);
698         /*
699          * If it's not the whole token, then it's not a match.
700          */
701         if (i < match_length) {
702             continue;
703         }
704         /*
705          * If it ties with another input, remember that.
706          */
707         if (i == length)
708             nmatches++;
709         /*
710          * If it matches the most so far, record that.
711          */
712         if (i > length) {
713             *match_value = slist->value;
714             nmatches = 1;
715             length = i;
716         }
717     }

```

```

719     return (nmatches);
720 }
721
722 static int32_t
723 strcnt(char *s1, char *s2)
724 {
725     int32_t i = 0;
726
727     while ((*s1 != '\0') && (*s1++ == *s2++))
728         i++;
729     return (i);
730 }
731
732 static uint64_t
733 str2sector(char *str)
734 {
735     int32_t mul_factor = 1;
736     char *s1, *s2, *base;
737     uint64_t num_sectors;
738     uint64_t size;
739
740     base = s2 = (char *)malloc(strlen(str) + 1);
741     if (s2 == NULL) {
742         PERROR("Malloc failed");
743         return (-1);
744     }
745     *s2 = '\0';
746
747     s1 = str;
748     while (*s1) {
749         if ((*s1 != 'x') && ((*s1 < 'A') || (*s1 > 'F')) &&
750             ((*s1 < 'a') || (*s1 > 'f')) && ((*s1 < '0') ||
751             (*s1 > '9')))) {
752             if (*s1 == 'G') {
753                 mul_factor = 1024*1024*1024;
754                 s1++;
755             } else if (*s1 == 'M') {
756                 mul_factor = 1024*1024;
757                 s1++;
758             } else if (*s1 == 'K') {
759                 mul_factor = 1024;
760                 s1++;
761             }
762             if ((*s1 != 'B') || ((*++s1) != NULL)) {
763                 (void) fprintf(stderr,
764                     gettext("Extra chars at the end\n"));
765                 free(base);
766                 return (-1);
767             }
768             break;
769         } else {
770             *s2++ = *s1++;
771             *s2 = '\0';
772         }
773     }
774     *s2 = NULL;
775
776     size = my_atoll(base);
777     if (!(mul_factor) || (size == -1)) {
778         free(base);
779         return (-1);
780     }
781     num_sectors = size * (uint64_t)mul_factor / 512;

```

```

785     free(base);
786     return (num_sectors);
787 }

790 int32_t
791 valid_slice_file(smedia_handle_t handle, int32_t fd, char *file_name,
792                struct extvtoc *vt)
793 {
794     struct stat status;
795     int32_t ret_val;
796     if (stat(file_name, &status)) {
797         PERROR(file_name);
798         return (-1);
799     }
800     (void) memset(vt, 0, sizeof(*vt));
801     /* Set default tag and flag */
802 #ifdef sparc
803     vt->v_part[0].p_tag = V_ROOT;
804     vt->v_part[1].p_tag = V_SWAP;
805     vt->v_part[2].p_tag = V_BACKUP;
806     vt->v_part[6].p_tag = V_USR;
807
808     vt->v_part[1].p_flag = V_UNMNT; /* Unmountable */
809     vt->v_part[2].p_flag = V_UNMNT; /* Unmountable */
810 #endif
811
812     ret_val = sup_prxfile(file_name, vt);
813     if (ret_val < 0)
814         return (-1);
815
816 #ifdef DEBUG
817 {
818     int32_t i;
819     for (i = 0; i < 8; i++) {
820         DPRINTF1("\npart %d\n", i);
821         DPRINTF1("\t start %llu", vt->v_part[i].p_start);
822         DPRINTF1("\t size %llu ", vt->v_part[i].p_size);
823         DPRINTF1("\t tag %d", vt->v_part[i].p_tag);
824         DPRINTF1("\t flag %d", vt->v_part[i].p_flag);
825     }
826 }
827 #endif /* DEBUG */
828     if (check_vtoc_sanity(handle, fd, vt) < 0) {
829         return (-1);
830     }
831 #ifdef DEBUG
832 {
833     int32_t i;
834     for (i = 0; i < 8; i++) {
835         DPRINTF1("\npart %d\n", i);
836         DPRINTF1("\t start %llu", vt->v_part[i].p_start);
837         DPRINTF1("\t size %llu ", vt->v_part[i].p_size);
838         DPRINTF1("\t tag %d", vt->v_part[i].p_tag);
839         DPRINTF1("\t flag %d", vt->v_part[i].p_flag);
840     }
841 }
842 #endif /* DEBUG */
843     return (0);
844 }
845
846 #define SWAP(a, b)    {diskaddr_t tmp; tmp = (a); (a) = (b); (b) = tmp; }
847
848 /*
849 * On x86 Solaris, the partitioning is done in two levels, fdisk and Solaris

```

```

850 * VTOC. Where as, on sparc solaris, it is only VTOC. On floppy and PCMCIA
851 * also it is assumed to be only VTOC, no fdisk.
852 *
853 * On sparc, the back up slice can cover the whole medium. But on x86
854 * (SCSI/ATAPI disks), the backup slice can cover the solaris partition
855 * in fdisk table.
856 * Following table describes how is it handled
857 * SPARC:
858 *     SCSI/ATAPI, floppy, pcmcia : don't check for fdisk.
859 *     DKIOCGGGEOM is sufficient.
860 * x86 : floppy, pcmcia : Don't check for fdisk. DKIOCGGGEOM is sufficient.
861 *     SCSI/ATAPI : Check for fdisk.
862 *     if not present, assume that the solaris
863 *     partition covers 100% of the medium
864 *     (minus one cylinder).
865 *
866 *     if present :
867 *         check for active solaris partition.
868 *         if not found, take the first solaris
869 *         partition.
870 *     If there are no solaris partitions, its an error, stop.
871 */

873 static int32_t
874 check_vtoc_sanity(smedia_handle_t handle, int32_t fd, struct extvtoc *vt)
875 {
876
877     int32_t i, j;
878     struct dk_geom dkg;
879     int32_t num_backup = 0;
880     diskaddr_t backup_size = 0;
881     struct part_struct {
882         diskaddr_t start;
883         diskaddr_t end;
884         int32_t num;
885     } part[NDKMAP];
886     diskaddr_t min_val;
887     int32_t min_slice, num_slices;
888     diskaddr_t media_size;
889     uint32_t cyl_size = 0;
890     uint32_t cyl_size;
891     int sparc_style = 0; /* sparc_style handling ? */
892     struct fdisk_info fdisk;
893     int sol_part;
894     int total_parts = 0;
895
896 #ifdef sparc
897     sparc_style = 1;
898 #endif /* sparc */
899
900     if ((med_info.sm_media_type == SM_FLOPPY) ||
901         (med_info.sm_media_type == SM_PCMCIA_MEM) ||
902         (med_info.sm_media_type == SM_PCMCIA_ATA) ||
903         (med_info.sm_media_type == SM SCSI_FLOPPY)) {
904         sparc_style = 1;
905     }
906
907     if (sparc_style) {
908         DPRINTF("sparc style true\n");
909         if (ioctl(fd, DKIOCGGGEOM, &dkg) < 0) {
910             PERROR("DKIOCGGGEOM Failed");
911             return (-1);
912         }
913         media_size = (diskaddr_t)dkg.dkg_ncyl * dkg.dkg_nhead *
914             dkg.dkg_nsect;
915         cyl_size = dkg.dkg_nhead * dkg.dkg_nsect;

```

```

915     }
916
917     if (!sparc_style) {
918     /*
919     * Try to get the fdisk information if available.
920     */
921         if (get_fdisk(handle, fd, 0, &fdisk) >= 0) {
922             /* fdisk table on disk */
923             sol_part = 0xFF;
924             for (i = 0; i < FD_NUMPART; i++) {
925                 if (fdisk.part[i].systid == SUNIXOS ||
926                     fdisk.part[i].systid == SUNIXOS2) {
927                     if (sol_part == 0xFF)
928                         sol_part = i;
929                     total_parts++;
930                     if (fdisk.part[i].bootid == ACTIVE)
931                         sol_part = i;
932                 }
933             }
934             if (sol_part == 0xFF) {
935                 /* No Solaris partition */
936
937                 (void) fprintf(stderr, gettext("No FDISK \
938 Solaris partition found!\n"));
939                 return (-1);
940             }
941             if (total_parts > 1)
942                 (void) fprintf(stderr, gettext("Multiple FDISK \
943 Solaris partitions found.\n"));
944             media_size = (diskaddr_t)fdisk.part[sol_part].numsect;
945
946             DPRINTF1("sol_part %d\n", sol_part);
947             DPRINTF1("media_size %llu\n", media_size);
948         } else {
949             DPRINTF("Didn't get fdisk\n");
950             /*
951             * No fdisk partition available. Assume a 100% Solaris.
952             * partition.
953             * Try getting disk geometry.
954             */
955             if (ioctl(fd, DKIOCGGEOM, &dkg) < 0)
956                 if (ioctl(fd, DKIOCG_PHYGEOM, &dkg) < 0) {
957                     DPRINTF("DKIOCG_PHYGEOM ioctl failed");
958                     return (-1);
959                 }
960             /* On x86 platform 1 cylinder is used for fdisk table */
961             dkg.dkg_ncyl = dkg.dkg_ncyl - 1;
962             media_size = (diskaddr_t)dkg.dkg_ncyl * dkg.dkg_nhead *
963                 dkg.dkg_nsect;
964         }
965     }
966
967 #ifdef DEBUG
968     DPRINTF1("Ncyl %d\n", dkg.dkg_ncyl);
969     DPRINTF1("nhead %d\n", dkg.dkg_nhead);
970     DPRINTF1("nsect %d\n", dkg.dkg_nsect);
971 #endif /* DEBUG */
972
973     if (media_size == 0) {
974         media_size = (uint32_t)med_info.sm_capacity;
975     }
976
977     (void) memset(&part, 0, sizeof(part));
978     for (i = 0, j = 0; i < NDKMAP; i++) {
979         if (vt->v_part[i].p_tag == V_BACKUP) {
980             if (vt->v_part[i].p_start != 0) {

```

```

981                 (void) fprintf(stderr,
982                     gettext(
983                         "Backup slice should start at sector 0\n"));
984                 return (-1);
985             }
986             backup_size = vt->v_part[i].p_size;
987             num_backup++;
988             continue;
989         }
990     }
991     if (vt->v_part[i].p_size) {
992         if (sparc_style) {
993             if (vt->v_part[i].p_start % cyl_size) {
994                 (void) fprintf(stderr,
995                     gettext(
996                         "Slice %d does not start on cylinder boundary\n"), i);
997                 (void) fprintf(stderr,
998                     gettext(
999                         "Cylinder size %d 512 byte sectors\n"), cyl_size);
1000                 return (-1);
1001             }
1002         }
1003         part[j].start = vt->v_part[i].p_start;
1004         part[j].end = vt->v_part[i].p_start +
1005             vt->v_part[i].p_size - 1;
1006         part[j].num = i;
1007         j++;
1008     }
1009 }
1010 if (num_backup > 1) {
1011     (void) fprintf(stderr,
1012         gettext("Maximum one backup slice is allowed\n"));
1013     (void) smedia_release_handle(handle);
1014     (void) close(fd);
1015     exit(1);
1016 }
1017 num_slices = j;
1018
1019 for (i = 0; i < num_slices; i++) {
1020     min_val = part[i].start;
1021     min_slice = i;
1022     for (j = i+1; j < num_slices; j++) {
1023         if (part[j].start < min_val) {
1024             min_val = part[j].start;
1025             min_slice = j;
1026         }
1027     }
1028     if (min_slice != i) {
1029         SWAP(part[i].start, part[min_slice].start)
1030         SWAP(part[i].end, part[min_slice].end)
1031         SWAP(part[i].num, part[min_slice].num)
1032     }
1033 }
1034
1035 #ifdef DEBUG
1036     for (i = 0; i < num_slices; i++) {
1037         DPRINTF4("\n %d (%d) : %llu, %llu", i, part[i].num,
1038             part[i].start, part[i].end);
1039     }
1040 #endif /* DEBUG */
1041
1042     if (backup_size > media_size) {
1043         if (sparc_style) {
1044             (void) fprintf(stderr,
1045                 gettext(
1046                     "Backup slice extends beyond size of media\n"));

```



```

1047         (void) fprintf(stderr,
1048             gettext("media size : %llu sectors \n"),
1049             media_size);
1050     } else {
1052         (void) fprintf(stderr,
1053             gettext("Backup slice extends beyond size of FDISK \
1054 Solaris partition\n"));
1055         (void) fprintf(stderr,
1056             gettext(
1057                 "FDISK Solaris partition size : %llu sectors \n"),
1058             media_size);
1059     }
1060     return (-1);
1061 }
1063 /*
1064  * If we have only backup slice return success here.
1065  */
1066 if (num_slices == 0)
1067     return (0);
1069 if (backup_size) {
1070     if (part[num_slices - 1].end > backup_size) {
1071         (void) fprintf(stderr,
1072             gettext("Slice %d extends beyond backup slice.\n"),
1073             part[num_slices - 1].num);
1074         return (-1);
1075     }
1076 } else {
1077     if (part[num_slices - 1].end > media_size) {
1078         if (sparc_style) {
1079             (void) fprintf(stderr,
1080                 gettext(
1081                     "Slice %d extends beyond media size\n"),
1082                 part[num_slices - 1].num);
1083             (void) fprintf(stderr,
1084                 gettext("media size : %llu sectors \n"),
1085                 media_size);
1086         } else {
1087             (void) fprintf(stderr,
1088                 gettext("Slice %d extends beyond FDISK"
1089                     " Solaris partition size\n"),
1090                 part[num_slices - 1].num);
1091             (void) fprintf(stderr, gettext(
1092                 "FDISK Solaris partition size : %llu "
1093                 "sectors \n"), media_size);
1094         }
1095         return (-1);
1096     }
1097 }
1101 for (i = 0; i < num_slices; i++) {
1102     if (i == 0)
1103         continue;
1104     if (part[i].start <= part[i-1].end) {
1105         (void) fprintf(stderr,
1106             gettext("Overlap between slices %d and %d\n"),
1107             part[i-1].num, part[i].num);
1108         (void) smedia_release_handle(handle);
1109         (void) close(fd);
1110         exit(1);
1111     }
1112 }

```

```

1114     return (0);
1115 }
_____unchanged_portion_omitted_

```

new/usr/src/cmd/rmvolmgr/vold.c

1

```
*****
29524 Tue Apr 22 07:25:49 2014
new/usr/src/cmd/rmvolmgr/vold.c
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 *
21 *
22 */
23 * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
24 * Use is subject to license terms.
25 *
26 * Copyright 2014 Andrew Stormont
27 * #endif /* ! codereview */
28 */
29 /*
30 * Vold compatibility for rmvolmgr: emulate old commands as well as
31 * action_filemgr.so to notify legacy apps via /tmp/removable pipes.
32 * A lot of this code is copied verbatim from vold sources.
33 *
34 * Here's the original description of action_filemgr.so:
35 *
36 * action_filemgr.so - filemgr interface routines for rmmount
37 *
38 * This shared object allows rmmount to communicate with filemgr.
39 * This is done by communicating over a named pipe that filemgr
40 * creates in directory NOTIFY_DIR. The name of the pipe must
41 * begin with NOTIFY_NAME. This source file contains #define
42 * compiler directives set the values of NOTIFY_DIR and NOTIFY_NAME.
43 *
44 * After a partition on a medium has been mounted as a result of
45 * either insertion or remounting of the medium, the action()
46 * method creates a file named with the symbolic name of the
47 * device in which the medium is inserted and the partition name
48 * (e.g. "jaz0-s2") in NOTIFY_DIR. The file consists of one text
49 * line containing a string naming the mount point of the partition,
50 * a string giving the raw device path to the partition, and a
51 * string naming the file system type on the partition. The action()
52 * method then sends a single character ('i' for insertion, 'r' for
53 * remounting) through the named pipe NOTIFY_NAME to tell filemgr to
54 * look for new files in NOTIFY_DIR.
55 *
56 * If a medium containing no mountable partitions is inserted
57 * or remounted in a device, the action() method creates a file
```

new/usr/src/cmd/rmvolmgr/vold.c

2

```
58 * named with the symbolic name of the device in NOTIFY_DIR.
59 * The file consists of one text line containing a string
60 * giving the symbolic name of the device and a string naming
61 * the reason that the medium couldn't be mounted. The action
62 * method then sends either an 'i' or an 'r' through the named
63 * pipe to tell filemgr to look for new files in NOTIFY_DIR.
64 *
65 * When a medium is ejected or unmounted, the action() method
66 * removes the files that were created in NOTIFY_DIR when the medium
67 * was inserted or remounted and sends a single character ('e' for
68 * ejection, 'u' for unmounting) through the named pipe.
69 *
70 * The following environment variables must be set before calling action():
71 *
72 *     VOLUME_ACTION          action that occurred (e.g. "insert", "eject")
73 *     VOLUME_SYMDEV         symbolic name (e.g. "cdrom0", "floppy1")
74 *     VOLUME_NAME           volume name (e.g. "unnamed_cdrom", "s2")
75 */
76
77
78 #include <stdio.h>
79 #include <stdlib.h>
80 #include <unistd.h>
81 #include <fcntl.h>
82 #include <string.h>
83 #include <strings.h>
84 #include <dirent.h>
85 #include <signal.h>
86 #include <errno.h>
87 #include <libintl.h>
88 #include <zone.h>
89 #include <pwd.h>
90 #include <sys/types.h>
91 #include <sys/stat.h>
92 #include <sys/dkio.h>
93 #include <sys/cdio.h>
94 #include <sys/vtoc.h>
95 #include <sys/param.h>
96 #include <sys/wait.h>
97 #include <libcontract.h>
98 #include <sys/contract/process.h>
99 #include <sys/ctfs.h>
100 #include <tsol/label.h>
101
102 #include "vold.h"
103 #include "rmm_common.h"
104
105 int          rmm_debug = 0;
106 boolean_t   rmm_vold_actions_enabled = B_FALSE;
107 boolean_t   rmm_vold_mountpoints_enabled = B_FALSE;
108
109 static char *prog_name = NULL;
110 static pid_t prog_pid = 0;
111 static int  system_labeled = 0;
112 static uid_t mnt_uid = (uid_t)-1;
113 static gid_t mnt_gid = (gid_t)-1;
114 static zoneid_t mnt_zoneid = -1;
115 static char mnt_zoneroot[MAXPATHLEN];
116 static char mnt_userdir[MAXPATHLEN];
117
118 /*
119 * Private attribute types and attributes.
120 */
121 static const char notify_characters[] = {
122     'e',
123     'i',
```

```
124     'r',
125     'u'
126 };

128 static const char *result_strings[] = {
129     "FALSE",
130     "TRUE"
131 };

133 #define NOTIFY_DIR      "/tmp/.removable"      /* dir where filemgr looks */
134 #define NOTIFY_NAME    "notify"              /* named pipe to talk over */

24 static void      volrmount_usage();
136 static void      volcheck_usage();
137 static int       vold_action(struct action_arg *aap);
138 static void      vold_update_mountpoints(struct action_arg *aap);
139 static char      *not_mountable(struct action_arg *aa);
140 static int       create_one_notify_file(char *fstype,
141                                         char *mount_point,
142                                         char *notify_file,
143                                         char *raw_partitionp,
144                                         char *reason,
145                                         char *symdev);
146 static int       create_notify_files(struct action_arg **aa);
147 static boolean_t notify_clients(action_t action, int do_notify);
148 static void      popdir(int fd);
149 static int       pushdir(const char *dir);
150 static boolean_t remove_notify_files(struct action_arg **aa);

152 /*
153  * should be called once from main()
154  */
155 /* ARGSUSED */
156 void
157 vold_init(int argc, char **argv)
158 {
159     system_labeled = is_system_labeled();
160 }
_____unchanged_portion_omitted_
```

new/usr/src/cmd/rmvolmgr/vold.h

1

```
*****
1974 Tue Apr 22 07:25:49 2014
new/usr/src/cmd/rmvolmgr/vold.h
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
*****
```

```
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 *
20 */
21 /*
22  * Copyright 2006 Sun Microsystems, Inc. All rights reserved.
23  * Use is subject to license terms.
24 *
25 * Copyright 2014 Andrew Stormont.
26 #endif /* ! codereview */
27 */

29 #ifndef _VOLD_H
30 #define _VOLD_H

24 #pragma ident "%Z%M% %I% %E% SMI"

32 #ifdef __cplusplus
33 extern "C" {
34 #endif

36 #include <libhal.h>

38 typedef enum {
39     EJECT,
40     INSERT,
41     REMOUNT,
42     UNMOUNT,
43     CLEAR_MOUNTS,
44     CLOSETRAY
45 } action_t;
_____unchanged_portion_omitted_____

59 extern int rmm_debug;
60 extern boolean_t rmm_vold_actions_enabled;
61 extern boolean_t rmm_vold_mountpoints_enabled;

63 void vold_init(int argc, char **argv);
64 int vold_postprocess(LibHalContext *hal_ctx, const char *udi,
65     struct action_arg *aap);
66 int vold_rmmount(int argc, char **argv);
```

new/usr/src/cmd/rmvolmgr/vold.h

2

```
61 int volrmmount(int argc, char **argv);
67 int volcheck(int argc, char **argv);

69 #ifdef __cplusplus
70 }
_____unchanged_portion_omitted_____
```

```

*****
13118 Tue Apr 22 07:25:49 2014
new/usr/src/man/man1/Makefile
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
*****
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet
9 # at http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2011, Richard Lowe
14 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
15 # Copyright 2014 Andrew Stormont.
16 #endif /* ! codereview */
17 #
18 #
19 include      $(SRC)/Makefile.master
20 #
21 MANSECT=     1
22 #
23 MANFILES=    acctcom.1      \
24              adb.1          \
25              addbib.1       \
26              alias.1        \
27              allocate.1     \
28              amt.1          \
29              appcert.1      \
30              apptrace.1     \
31              apropos.1      \
32              ar.1           \
33              arch.1         \
34              asa.1          \
35              at.1           \
36              atq.1          \
37              atrm.1         \
38              audioconvert.1 \
39              audiocvt.1     \
40              audioplay.1    \
41              audiorecord.1  \
42              audiotest.1    \
43              auths.1        \
44              awk.1          \
45              banner.1       \
46              basename.1    \
47              bc.1           \
48              bdiff.1        \
49              bfs.1          \
50              break.1        \
51              builtin.1     \
52              cal.1          \
53              calendar.1    \
54              cancel.1      \
55              cat.1          \
56              cd.1           \
57              cdrw.1         \
58              checknr.1     \
59              chgrp.1

```

```

60              chkey.1        \
61              chmod.1        \
62              chown.1        \
63              ckdate.1       \
64              ckgid.1        \
65              ckint.1        \
66              ckitem.1       \
67              ckkeywd.1      \
68              ckpath.1       \
69              ckrange.1     \
70              ckstr.1        \
71              cksum.1        \
72              cktime.1       \
73              ckuid.1        \
74              ckyorn.1       \
75              clear.1        \
76              cmp.1          \
77              col.1          \
78              comm.1         \
79              command.1     \
80              compress.1    \
81              cp.1           \
82              cpio.1         \
83              cputrack.1     \
84              crle.1         \
85              crontab.1     \
86              crypt.1        \
87              csh.1          \
88              csplit.1       \
89              ctags.1        \
90              ctrun.1        \
91              ctstat.1       \
92              ctwatch.1      \
93              cut.1          \
94              date.1         \
95              dc.1           \
96              deallocate.1   \
97              deroff.1       \
98              dhcinfo.1     \
99              diff.1         \
100             diff3.1        \
101             diffmk.1       \
102             digest.1       \
103             dircmp.1       \
104             dis.1          \
105             disown.1       \
106             dispgid.1      \
107             dispuid.1      \
108             dos2unix.1     \
109             download.1     \
110             dpost.1        \
111             du.1           \
112             dump.1         \
113             dumpcs.1       \
114             echo.1         \
115             ed.1           \
116             egrep.1        \
117             eject.1        \
118             elfdump.1      \
119             elfedit.1      \
120             elfsign.1      \
121             elfwrap.1      \
122             enable.1       \
123             encrypt.1      \
124             enhance.1     \
125             env.1

```

```

126          eqn.1          \
127          exec.1         \
128          exit.1         \
129          expand.1        \
130          expr.1          \
131          exstr.1         \
132          factor.1        \
133          fdformat.1      \
134          fgrep.1         \
135          file.1           \
136          filebench.1     \
137          filesync.1      \
138          find.1           \
139          finger.1        \
140          fmt.1           \
141          fmtmsg.1        \
142          fold.1          \
143          ftp.1           \
144          ftpcount.1      \
145          ftpwho.1        \
146          gcore.1         \
147          gencat.1        \
148          genmsg.1        \
149          getconf.1       \
150          getfacl.1       \
151          getlabel.1      \
152          getopt.1        \
153          getoptcv.1      \
154          getopts.1       \
155          gettext.1       \
156          gettxt.1        \
157          getzonepath.1   \
158          glob.1          \
159          gprof.1         \
160          grep.1          \
161          groups.1        \
162          hash.1          \
163          head.1          \
164          history.1       \
165          hostid.1        \
166          hostname.1     \
167          iconv.1         \
168          indxib.1        \
169          Intro.1         \
170          ipcrm.1         \
171          ipcs.1          \
172          isainfo.1       \
173          isalist.1       \
174          jobs.1          \
175          join.1          \
176          kbd.1           \
177          kdestroy.1     \
178          keylogin.1      \
179          keylogout.1     \
180          kill.1          \
181          kinit.1         \
182          klist.1         \
183          kmdb.1          \
184          kmfcfg.1        \
185          kpasswd.1       \
186          krb5-config.1   \
187          ksh93.1        \
188          ktutil.1        \
189          lari.1          \
190          last.1          \
191          lastcomm.1     \

```

```

192          ld.1           \
193          ldap.1         \
194          ldapdelete.1   \
195          ldaplist.1     \
196          ldapmodify.1   \
197          ldapmodrdn.1   \
198          ldapsearch.1   \
199          ldd.1          \
200          ld.so.1.1      \
201          let.1          \
202          lex.1          \
203          lgrpinfo.1     \
204          limit.1        \
205          line.1         \
206          list_devices.1 \
207          listusers.1    \
208          ln.1           \
209          loadkeys.1     \
210          locale.1       \
211          localedef.1    \
212          logger.1       \
213          login.1        \
214          logname.1      \
215          logout.1       \
216          look.1         \
217          lookbib.1      \
218          lorder.1       \
219          lp.1           \
220          lpstat.1       \
221          ls.1           \
222          m4.1           \
223          mac.1          \
224          mach.1         \
225          machid.1       \
226          adv.so.1.1     \
227          mail.1         \
228          mailcompat.1   \
229          mailq.1        \
230          mailstats.1    \
231          mailx.1        \
232          makekey.1      \
233          man.1          \
234          mconnect.1     \
235          mcs.1          \
236          mdb.1          \
237          msg.1          \
238          mkdir.1        \
239          mkmsgs.1       \
240          mktemp.1       \
241          moe.1          \
242          more.1         \
243          mpss.so.1.1    \
244          msgcc.1        \
245          msgcpp.1       \
246          msgcv.1        \
247          msgfmt.1       \
248          msggen.1       \
249          msgget.1       \
250          mt.1           \
251          mv.1           \
252          nawk.1         \
253          nc.1           \
254          nca.1          \
255          ncab2cl.1      \
256          ncakmod.1     \
257          newform.1     \

```

```

258 newgrp.1 \
259 news.1 \
260 newtask.1 \
261 nice.1 \
262 nl.1 \
263 nm.1 \
264 nohup.1 \
265 nroff.1 \
266 od.1 \
267 on.1 \
268 optisa.1 \
269 pack.1 \
270 pagesize.1 \
271 pargs.1 \
272 passwd.1 \
273 paste.1 \
274 pathchk.1 \
275 pax.1 \
276 pfexec.1 \
277 pg.1 \
278 pgrep.1 \
279 pkginfo.1 \
280 pkgmk.1 \
281 pkgparam.1 \
282 pkgproto.1 \
283 pkgtrans.1 \
284 pktool.1 \
285 plabel.1 \
286 plgrp.1 \
287 plimit.1 \
288 pmadvise.1 \
289 pmap.1 \
290 postio.1 \
291 postprint.1 \
292 postreverse.1 \
293 ppgsz.1 \
294 ppriv.1 \
295 pr.1 \
296 praliases.1 \
297 prctl.1 \
298 preap.1 \
299 prex.1 \
300 print.1 \
301 printf.1 \
302 priocntl.1 \
303 proc.1 \
304 prof.1 \
305 profiles.1 \
306 projects.1 \
307 ps.1 \
308 ptree.1 \
309 pvs.1 \
310 pwd.1 \
311 ranlib.1 \
312 rcapstat.1 \
313 rcp.1 \
314 rdist.1 \
315 read.1 \
316 readonly.1 \
317 refer.1 \
318 regcmp.1 \
319 renice.1 \
320 rev.1 \
321 rlogin.1 \
322 rm.1 \
323 rmformat.1 \

```

```

324 rmount.1 \
325 roffbib.1 \
326 roles.1 \
327 rpcgen.1 \
328 rsh.1 \
329 runat.1 \
330 rup.1 \
331 ruptime.1 \
332 rusers.1 \
333 rwho.1 \
334 sar.1 \
335 scp.1 \
336 script.1 \
337 sdiff.1 \
338 sed.1 \
339 set.1 \
340 setfacl.1 \
341 setlabel.1 \
342 setpgrp.1 \
343 sftp.1 \
344 shcomp.1 \
345 shell_builtins.1 \
346 shift.1 \
347 size.1 \
348 sleep.1 \
349 smbutil.1 \
350 soelim.1 \
351 sort.1 \
352 sortbib.1 \
353 sotruss.1 \
354 spell.1 \
355 split.1 \
356 srchtxt.1 \
357 ssh.1 \
358 ssh-add.1 \
359 ssh-agent.1 \
360 ssh-http-proxy-connect.1 \
361 ssh-keygen.1 \
362 ssh-keyscan.1 \
363 ssh-socks5-proxy-connect.1 \
364 strchg.1 \
365 strings.1 \
366 strip.1 \
367 stty.1 \
368 sum.1 \
369 suspend.1 \
370 svcprop.1 \
371 svcs.1 \
372 symorder.1 \
373 sys-suspend.1 \
374 tabs.1 \
375 tail.1 \
376 talk.1 \
377 tar.1 \
378 tbl.1 \
379 tcopy.1 \
380 tee.1 \
381 telnet.1 \
382 test.1 \
383 tftp.1 \
384 time.1 \
385 times.1 \
386 timex.1 \
387 tip.1 \
388 tnfdump.1 \
389 tnfxtract.1 \

```

```

390 touch.1 \
391 tput.1 \
392 tr.1 \
393 trap.1 \
394 troff.1 \
395 true.1 \
396 truss.1 \
397 tsort.1 \
398 tty.1 \
399 type.1 \
400 typeset.1 \
401 ul.1 \
402 umask.1 \
403 uname.1 \
404 unifdef.1 \
405 uniq.1 \
406 units.1 \
407 unix2dos.1 \
408 uptime.1 \
409 vacation.1 \
410 vgrind.1 \
411 volcheck.1 \
    15 volrmount.1 \
412 w.1 \
413 wait.1 \
414 wc.1 \
415 whatis.1 \
416 which.1 \
417 who.1 \
418 whocalls.1 \
419 whois.1 \
420 write.1 \
421 xargs.1 \
422 xgettext.1 \
423 xstr.1 \
424 yacc.1 \
425 yes.1 \
426 ypcat.1 \
427 ypmatch.1 \
428 yppasswd.1 \
429 ypwhich.1 \
430 zlogin.1 \
431 zone.1 \

433 MANLINKS= batch.1 \
434 bg.1 \
435 case.1 \
436 chdir.1 \
437 checkeg.1 \
438 continue.1 \
439 decrypt.1 \
440 dirname.1 \
441 dirs.1 \
442 disable.1 \
443 dumpkeys.1 \
444 edit.1 \
445 errange.1 \
446 errdate.1 \
447 errgid.1 \
448 errint.1 \
449 erritem.1 \
450 errpath.1 \
451 errstr.1 \
452 errtime.1 \
453 erruid.1 \
454 erryorn.1 \

```

```

455 eval.1 \
456 export.1 \
457 false.1 \
458 fc.1 \
459 fg.1 \
460 for.1 \
461 foreach.1 \
462 function.1 \
463 goto.1 \
464 hashcheck.1 \
465 hashmake.1 \
466 hashstat.1 \
467 helpdate.1 \
468 helpgid.1 \
469 helpint.1 \
470 helpitem.1 \
471 helppath.1 \
472 helprange.1 \
473 helpstr.1 \
474 helptime.1 \
475 helpuid.1 \
476 helpyorn.1 \
477 hist.1 \
478 i286.1 \
479 i386.1 \
480 i486.1 \
481 i860.1 \
482 iAPX286.1 \
483 if.1 \
484 intro.1 \
485 jsh.1 \
486 ksh.1 \
487 ldapadd.1 \
488 neqn.1 \
489 notify.1 \
490 onintr.1 \
491 page.1 \
492 pcat.1 \
493 pcred.1 \
494 pdpl1.1 \
495 pfcsh.1 \
496 pfiles.1 \
497 pfksh.1 \
498 pflags.1 \
499 pfs.1 \
500 pkill.1 \
501 pldd.1 \
502 popd.1 \
503 prun.1 \
504 psig.1 \
505 pstack.1 \
506 pstop.1 \
507 ptime.1 \
508 pushd.1 \
509 pwait.1 \
510 pwdx.1 \
511 red.1 \
512 rehash.1 \
513 remote_shell.1 \
514 remsh.1 \
515 repeat.1 \
516 return.1 \
517 rksh.1 \
518 rksh93.1 \
519 rmail.1 \
520 rmdir.1 \

```



```

521          rmumount.1  \
522          select.1    \
523          setenv.1    \
524          settime.1   \
525          sh.1        \
526          snca.1      \
527          source.1    \
528          sparc.1     \
529          spellin.1   \
530          stop.1      \
531          strconf.1   \
532          sun.1       \
533          switch.1    \
534          u370.1      \
535          u3b.1       \
536          u3b15.1     \
537          u3b2.1      \
538          u3b5.1      \
539          ulimit.1    \
540          unalias.1   \
541          uncompress.1 \
542          unexpand.1 \
543          unhash.1    \
544          unlimit.1   \
545          unpack.1    \
546          unset.1     \
547          unsetenv.1 \
548          until.1     \
549          valdate.1   \
550          valgid.1    \
551          valint.1    \
552          valpath.1   \
553          valrange.1  \
554          valstr.1    \
555          valtime.1   \
556          valuid.1    \
557          valyorn.1   \
558          vax.1       \
559          vedit.1     \
560          whence.1    \
561          while.1     \
562          zcat.1      \

564 intro.1      := LINKSRC = Intro.1
566 unalias.1    := LINKSRC = alias.1
568 batch.1      := LINKSRC = at.1
570 dirname.1    := LINKSRC = basename.1
572 continue.1   := LINKSRC = break.1
574 chdir.1      := LINKSRC = cd.1
575 dirs.1       := LINKSRC = cd.1
576 popd.1       := LINKSRC = cd.1
577 pushd.1      := LINKSRC = cd.1

579 errdate.1     := LINKSRC = ckdate.1
580 helpdate.1    := LINKSRC = ckdate.1
581 valdate.1     := LINKSRC = ckdate.1

583 errgid.1      := LINKSRC = ckgid.1
584 helpgid.1     := LINKSRC = ckgid.1
585 valgid.1      := LINKSRC = ckgid.1

```

```

587 errint.1      := LINKSRC = ckint.1
588 helpint.1     := LINKSRC = ckint.1
589 valint.1      := LINKSRC = ckint.1

591 erritem.1     := LINKSRC = ckitem.1
592 helpitem.1    := LINKSRC = ckitem.1

594 errpath.1     := LINKSRC = ckpath.1
595 helppath.1    := LINKSRC = ckpath.1
596 valpath.1     := LINKSRC = ckpath.1

598 errrange.1    := LINKSRC = ckrange.1
599 helprange.1   := LINKSRC = ckrange.1
600 valrange.1    := LINKSRC = ckrange.1

602 errstr.1      := LINKSRC = ckstr.1
603 helpstr.1     := LINKSRC = ckstr.1
604 valstr.1      := LINKSRC = ckstr.1

606 errtime.1     := LINKSRC = cktime.1
607 helptime.1    := LINKSRC = cktime.1
608 valtime.1     := LINKSRC = cktime.1

610 erruid.1      := LINKSRC = ckuid.1
611 helpuid.1     := LINKSRC = ckuid.1
612 valuid.1      := LINKSRC = ckuid.1

614 erryorn.1     := LINKSRC = ckyorn.1
615 helpyorn.1    := LINKSRC = ckyorn.1
616 valyorn.1     := LINKSRC = ckyorn.1

618 uncompress.1  := LINKSRC = compress.1
619 zcat.1        := LINKSRC = compress.1

621 red.1         := LINKSRC = ed.1

623 disable.1    := LINKSRC = enable.1

625 decrypt.1    := LINKSRC = encrypt.1

627 checkeq.1    := LINKSRC = eqn.1
628 neqn.1       := LINKSRC = eqn.1

630 eval.1       := LINKSRC = exec.1
631 source.1     := LINKSRC = exec.1

633 goto.1      := LINKSRC = exit.1
634 return.1     := LINKSRC = exit.1

636 unexpand.1  := LINKSRC = expand.1

638 hashstat.1  := LINKSRC = hash.1
639 rehash.1    := LINKSRC = hash.1
640 unhash.1    := LINKSRC = hash.1

642 fc.1        := LINKSRC = history.1
643 hist.1      := LINKSRC = history.1

645 bg.1        := LINKSRC = jobs.1
646 fg.1        := LINKSRC = jobs.1
647 notify.1    := LINKSRC = jobs.1
648 stop.1      := LINKSRC = jobs.1

650 jsh.1       := LINKSRC = ksh93.1
651 ksh.1       := LINKSRC = ksh93.1
652 rksh.1      := LINKSRC = ksh93.1

```

new/usr/src/man/man1/Makefile

11

```

653 rksh93.1      := LINKSRC = ksh93.1
654 sh.1          := LINKSRC = ksh93.1

656 ldapadd.1     := LINKSRC = ldapmodify.1

658 ulimit.1      := LINKSRC = limit.1
659 unlimit.1     := LINKSRC = limit.1

661 dumpkeys.1    := LINKSRC = loadkeys.1

663 i286.1        := LINKSRC = machid.1
664 i386.1        := LINKSRC = machid.1
665 i486.1        := LINKSRC = machid.1
666 i860.1        := LINKSRC = machid.1
667 iAPX286.1     := LINKSRC = machid.1
668 pdp11.1       := LINKSRC = machid.1
669 sparc.1       := LINKSRC = machid.1
670 sun.1         := LINKSRC = machid.1
671 u370.1        := LINKSRC = machid.1
672 u3b.1         := LINKSRC = machid.1
673 u3b15.1       := LINKSRC = machid.1
674 u3b2.1        := LINKSRC = machid.1
675 u3b5.1        := LINKSRC = machid.1
676 vax.1         := LINKSRC = machid.1

678 rmail.1       := LINKSRC = mail.1

680 page.1        := LINKSRC = more.1

682 snca.1        := LINKSRC = nca.1

684 pcat.1        := LINKSRC = pack.1
685 unpack.1     := LINKSRC = pack.1

687 pfcsh.1       := LINKSRC = pfexec.1
688 pfksh.1       := LINKSRC = pfexec.1
689 pfsh.1        := LINKSRC = pfexec.1

691 pkill.1       := LINKSRC = pgrep.1

693 pcred.1       := LINKSRC = proc.1
694 pfiles.1      := LINKSRC = proc.1
695 pflags.1      := LINKSRC = proc.1
696 pldd.1        := LINKSRC = proc.1
697 prun.1        := LINKSRC = proc.1
698 psig.1        := LINKSRC = proc.1
699 pstack.1      := LINKSRC = proc.1
700 pstop.1       := LINKSRC = proc.1
701 ptime.1       := LINKSRC = proc.1
702 pwait.1       := LINKSRC = proc.1
703 pwdx.1        := LINKSRC = proc.1

705 rmdir.1       := LINKSRC = rm.1

707 rmumount.1    := LINKSRC = rmmount.1

709 remote_shell.1 := LINKSRC = rsh.1
710 remsh.1       := LINKSRC = rsh.1

712 export.1      := LINKSRC = set.1
713 setenv.1      := LINKSRC = set.1
714 unset.1       := LINKSRC = set.1
715 unsetenv.1   := LINKSRC = set.1

717 case.1        := LINKSRC = shell_builtins.1
718 for.1         := LINKSRC = shell_builtins.1

```

new/usr/src/man/man1/Makefile

12

```

719 foreach.1     := LINKSRC = shell_builtins.1
720 function.1    := LINKSRC = shell_builtins.1
721 if.1          := LINKSRC = shell_builtins.1
722 repeat.1      := LINKSRC = shell_builtins.1
723 select.1      := LINKSRC = shell_builtins.1
724 switch.1      := LINKSRC = shell_builtins.1
725 until.1       := LINKSRC = shell_builtins.1
726 while.1       := LINKSRC = shell_builtins.1

728 hashcheck.1   := LINKSRC = spell.1
729 hashmake.1    := LINKSRC = spell.1
730 spellin.1     := LINKSRC = spell.1

732 strconf.1     := LINKSRC = strchg.1

734 settime.1     := LINKSRC = touch.1

736 onintr.1      := LINKSRC = trap.1

738 false.1       := LINKSRC = true.1

740 whence.1      := LINKSRC = typeset.1

742 # Links to usr/has/man

744 edit.1        := LINKSRC = ../../../../has/man/man1has/edit.lhas

746 vedit.1       := LINKSRC = ../../../../has/man/man1has/vi.lhas

748 .KEEP_STATE:

750 include        $(SRC)/man/Makefile.man

752 install:      $(ROOTMANFILES) $(ROOTMANLINKS)

```

```

*****
11823 Tue Apr 22 07:25:49 2014
new/usr/src/man/man1/fdformat.1
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
*****
1 \" te
2 .\" Copyright 2001, Sun Microsystems, Inc All Rights Reserved
3 .\" Copyright 2014 Andrew Stormont.
4 #endif /* !codereview */
5 .\" The contents of this file are subject to the terms of the Common Development
6 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
7 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
8 .TH FDFORMAT 1 \"May 2, 2014\"
9 .TH FDFORMAT 1 \"Feb 28, 2007\"
10 .SH NAME
11 fdformat - format floppy diskette or PCMCIA memory card
12 .SH SYNOPSIS
13 .LP
14 \fBfdformat\fR [\fB-dDeEfHlLmMUqvX\fR] [\fB-b\fR \fIlabel\fR] [\fB-B\fR \fIfilen
15 [\fB-t\fR \fIidostype\fR] [\fB-v\fR \fIdevname\fR]
16 .fi

18 .SH DESCRIPTION
19 .sp
20 .LP
21 The \fBfdformat\fR utility has been superseded by \fBbrmformat\fR(1), which
22 provides most but not all of \fBfdformat\fR's functionality.
23 .sp
24 .LP
25 \fBfdformat\fR is used to format diskettes and \fBPCMCIA\fR memory cards. All
26 new blank diskettes or \fBPCMCIA\fR memory cards must be formatted before they
27 can be used.
28 .sp
29 .LP
30 \fBfdformat\fR formats and verifies the media and indicates whether any bad
31 sectors were encountered. All existing data on the diskette or \fBPCMCIA\fR
32 memory card, if any, is destroyed by formatting. If no device name is given,
33 \fBfdformat\fR uses the diskette as a default.
34 .sp
35 .LP
36 By default, \fBfdformat\fR uses the configured capacity of the drive to format
37 the diskette. A \fB3.5\fR inch high-density drive uses diskettes with a
38 formatted capacity of \fB1.44MB\fR. A \fB5.25\fR inch high-density drive uses
39 diskettes with a formatted capacity of \fB1.2MB\fR. In either case, a density
40 option does not have to be specified to \fBfdformat\fR. However, a density
41 option must be specified when using a diskette with a lower capacity than the
42 drive's default. Use the \fB-H\fR option to format high-density diskettes
43 (\fB1.44MB\fR capacity) in an extra-high-density (\fBED\fR) drive. Use the
44 \fB-D\fR option, the \fB-l\fR option, or the \fB-L\fR option to format double-
45 density (or low-density) diskettes (\fB720KB\fR capacity) in an \fBHDD\fR or
46 \fBED\fR drive. To format medium-density diskettes (\fB1.2MB\fR capacity), use
47 the \fB-M\fR option with \fB-t\fR \fBnec\fR (this is the same as using the
48 \fB-m\fR option with \fB-t\fR \fBnec\fR).
49 .sp
50 .LP
51 Extended density uses double-sided, extended-density or extra-high-density
52 (\fBDS\fR/\fBED\fR) diskettes. Medium and high densities use the same media:
53 double-sided, high-density (\fBDS\fR/\fBHD\fR) diskettes. Double (low) density
54 uses double-sided, double-density (\fBDS\fR/\fBDD\fR
55 .sp
56 .LP
57 \fBD\fR) diskettes. Substituting diskettes of one density for diskettes of
58 either a higher or lower density generally does not work. Data integrity cannot

```

```

59 be assured whenever a diskette is formatted to a capacity not matching its
60 density.
61 .sp
62 .LP
63 A \fBPCMCIA\fR memory card with densities from \fB512KB\fR to \fB64MB\fR may be
64 formatted.
65 .sp
66 .LP
67 \fBfdformat\fR writes new identification and data fields for each sector on all
68 tracks unless the \fB-x\fR option is specified. For diskettes, each sector is
69 verified if the \fB-v\fR option is specified.
70 .sp
71 .LP
72 After formatting and verifying, \fBfdformat\fR writes an operating-system label
73 on block \fB0\fR. Use the \fB-t\fR \fBdos\fR option (same as the \fB-d\fR
74 option) to put an \fBMS-DOS\fR file system on the diskette or \fBPCMCIA\fR
75 memory card after the format is done. Use the \fB-t\fR \fBnec\fR option with
76 the \fB-M\fR option (same as the \fB-m\fR option) to put an \fBNEC-DOS\fR file
77 system on a diskette. Otherwise, \fBfdformat\fR writes a \fBSunOS\fR label in
78 block \fB0\fR.
79 .SH OPTIONS
80 .sp
81 .LP
82 The following options are supported:
83 .sp
84 .ne 2
85 .na
86 \fB-b\fR \fIlabel\fR
87 .ad
88 .RS 15n
89 Labels the media with volume label. A SunOS volume label is restricted to 8
90 characters. A \fBDOS\fR volume label is restricted to 11 upper-case characters.
91 .RE

93 .sp
94 .ne 2
95 .na
96 \fB-B\fR \fIfilename\fR
97 .ad
98 .RS 15n
99 Installs special boot loader in filename on an \fBMS-DOS\fR diskette. This
100 option is only meaningful when the \fB-d\fR option (or \fB-t\fR \fBdos\fR) is
101 also specified.
102 .RE

104 .sp
105 .ne 2
106 .na
107 \fB-B-D\fR
108 .ad
109 .RS 15n
110 Formats a \fB720KB\fR (3.5 inch) or \fB360KB\fR (5.25 inch) double-density
111 diskette (same as the \fB-l\fR or \fB-L\fR options). This is the default for
112 double-density type drives. It is needed if the drive is a high- or
113 extended-density type.
114 .RE

116 .sp
117 .ne 2
118 .na
119 \fB-b-e\fR
120 .ad
121 .RS 15n
122 Ejects the diskette when done. This feature is not available on all systems.
123 .RE

```

```

125 .sp
126 .ne 2
127 .na
128 \fB\fB-E\fR\fR
129 .ad
130 .RS 15n
131 Formats a \fB2.88MB\fR (3.5 inch) extended-density diskette. This is the
132 default for extended-density type drives.
133 .RE

135 .sp
136 .ne 2
137 .na
138 \fB\fB-f\fR\fR
139 .ad
140 .RS 15n
141 Forces formatting, that is, this option does not ask for confirmation before
142 starting format.
143 .RE

145 .sp
146 .ne 2
147 .na
148 \fB\fB-H\fR\fR
149 .ad
150 .RS 15n
151 Formats a \fB1.44MB\fR (3.5 inch) or \fB1.2MB\fR (5.25 inch) high-density
152 diskette. This is the default for high-density type drives; it is needed if the
153 drive is the extended-density type.
154 .RE

156 .sp
157 .ne 2
158 .na
159 \fB\fB-M\fR\fR
160 .ad
161 .RS 15n
162 Writes a \fB1.2MB\fR (3.5 inch) medium-density format on a high-density
163 diskette (use only with the -t nec option). This is the same as using \fB-m\fR.
164 .sp
165 This feature is not available on all systems.
166 .RE

168 .sp
169 .ne 2
170 .na
171 \fB\fB-q\fR\fR
172 .ad
173 .RS 15n
174 Quiet; does not print status messages.
175 .RE

177 .sp
178 .ne 2
179 .na
180 \fB\fB-t\fR \fBdos\fR\fR
181 .ad
182 .RS 15n
183 Installs an \fBMS-DOS\fR file system and boot sector formatting. This is
184 equivalent to the \fBDOS\fR format command or the \fB-d\fR option.
185 .RE

187 .sp
188 .ne 2
189 .na
190 \fB\fB-t\fR \fBnec\fR\fR

```

```

191 .ad
192 .RS 15n
193 Installs an \fBNEC-DOS\fR file system and boot sector on the disk after
194 formatting. This should be used only with the \fB-M\fR option. This feature is
195 not available on all systems.
196 .RE

198 .sp
199 .ne 2
200 .na
201 \fB\fB-U\fR\fR
202 .ad
203 .RS 15n
204 Performs \fBumount\fR on any file systems and then formats. See
205 \fBmount\fR(1M).
206 .RE

208 .sp
209 .ne 2
210 .na
211 \fB\fB-v\fR\fR
212 .ad
213 .RS 15n
214 Verifies each block of the diskette after the format.
215 .RE

217 .sp
218 .ne 2
219 .na
220 \fB\fB-x\fR\fR
221 .ad
222 .RS 15n
223 Skips the format and only writes a SunOS label or an \fBMS-DOS\fR file system.
224 .RE

226 .SH OPERANDS
227 .sp
228 .LP
229 The following operands are supported:
230 .sp
231 .ne 2
232 .na
233 \fB\fIdevname\fR\fR
234 .ad
235 .RS 11n
236 Replaces \fIdevname\fR with \fBrdiskette0\fR (systems without volume
237 management) or \fBfloppy0\fR (systems with volume management) to use the first
238 drive or \fBrdiskette1\fR (systems without volume management) or \fBfloppy1\fR
239 (systems with volume management) to use the second drive. If \fIdevname\fR is
240 omitted, the first drive, if one exists, is used. For \fBPCMCIA\fR memory
241 cards, replace \fIdevname\fR with the device name for the \fBPCMCIA\fR memory
242 card which resides in \fB/dev/rdisk/c\fIN\fRt\fIN\fRd\fIN\fRs\fIN\fR or
243 \fB/dev/dsk/c\fIN\fRt\fIN\fRd\fIN\fRs\fIN\fR. If \fIdevname\fR is omitted, the
244 default diskette drive, if one exists, is used.
245 .sp
246 If devname is omitted, the default diskette drive, if one exists, will be used.
247 \fIN\fR represents a decimal number and can be specified as follows:
248 .sp
249 .ne 2
250 .na
251 \fBc\fIN\fR\fR
252 .ad
253 .RS 6n
254 Controller \fIN\fR
255 .RE

```

```

257 .sp
258 .ne 2
259 .na
260 \fBt\fIN\fR\fR
261 .ad
262 .RS 6n
263 Technology type \fIN\fR:
264 .sp
265 .in +2
266 .nf

268 0x1      ROM
269 0x2      OTPROM
270 0x3      EPROM
271 0x4      EEPROM
272 0x5      FLASH
273 0x6      SRAM
274 0x7      DRAM
275 .fi
276 .in -2
277 .sp

279 .RE

281 .sp
282 .ne 2
283 .na
284 \fBd\fIN\fR\fR
285 .ad
286 .RS 6n
287 Technology region in type \fIN\fR.
288 .RE

290 .sp
291 .ne 2
292 .na
293 \fBs\fIN\fR\fR
294 .ad
295 .RS 6n
296 Slice \fIN\fR.
297 .RE

299 The following options are provided for compatibility with previous versions of
300 \fBfdformat\fR. Their use is discouraged.
301 .sp
302 .ne 2
303 .na
304 \fB\FB-d\fR\fR
305 .ad
306 .RS 6n
307 Formats an \fBMS-DOS\fR floppy diskette or \fBPCMCIA\fR memory card (same as
308 \fB-t\fR \fBdos\fR). This is equivalent to the \fBMS-DOS FORMAT\fR command.
309 .RE

311 .sp
312 .ne 2
313 .na
314 \fB\FB-l\fR\fR
315 .ad
316 .RS 6n
317 Formats a \fB720KB\fR (3.5 inch) or \fB360KB\fR (5.25 inch) double-density
318 diskette (same as \fB-D\fR or \fB-L\fR). This is the default for double-density
319 type drives; it is needed if the drive is the high- or extended-density type.
320 .RE

322 .sp

```

```

323 .ne 2
324 .na
325 \fB\FB-L\fR\fR
326 .ad
327 .RS 6n
328 Formats a \fB720KB\fR (3.5 inch) or \fB360KB\fR (5.25 inch) double-density
329 diskette (same as \fB-l\fR or \fB-D\fR). This is the default for double-density
330 type drives.
331 .RE

333 .sp
334 .ne 2
335 .na
336 \fB\FB-m\fR\fR
337 .ad
338 .RS 6n
339 Writes a \fB1.2 MB\fR (3.5 inch) medium- density format on a high-density
340 diskette (use only with the \fB-t\fR \fBnec\fR option). This is the same as
341 using \fB-M\fR. This feature is not available on all systems.
342 .RE

344 .RE

346 .SH FILES
347 .sp
348 .ne 2
349 .na
350 \fB/dev/diskette0\fR
351 .ad
352 .RS 24n
353 Directory providing block device access for the media in floppy drive \fB0\fR.
354 .RE

356 .sp
357 .ne 2
358 .na
359 \fB/dev/diskette0\fR
360 .ad
361 .RS 24n
362 Directory providing character device access for the media in floppy drive
363 \fB0\fR.
364 .RE

366 .sp
367 .ne 2
368 .na
369 \fB/dev/aliases/floppy0\fR
370 .ad
371 .RS 24n
372 Symbolic link to the character device for the media in floppy drive \fB0\fR.
373 .RE

375 .sp
376 .ne 2
377 .na
378 \fB/dev/rdiskette\fR
379 .ad
380 .RS 24n
381 Directory providing character device access for the media in the primary floppy
382 drive, usually drive \fB0\fR.
383 .RE

385 .sp
386 .ne 2
387 .na
388 \fB/dev/dsk/c\fIN\fRt\fIN\fRd\fIN\fRs\fIN\fR\fR

```

```

389 .ad
390 .RS 24n
391 Directory providing block device access for the \fBPCMCIA\fR memory card. See
392 OPERANDS for a description of \fIN\fR.
393 .RE

395 .sp
396 .ne 2
397 .na
398 \fB/dev/rdisk/c\fIN\fRt\fIN\fRd\fIN\fRs\fIN\fR\fR
399 .ad
400 .RS 24n
401 Directory providing character device access for the \fBPCMCIA\fR memory card.
402 See OPERANDS for a description of \fIN\fR.
403 .RE

405 .sp
406 .ne 2
407 .na
408 \fB/dev/aliases/pcmем\fIS\fR\fR
409 .ad
410 .RS 24n
411 Symbolic link to the character device for the \fBPCMCIA\fR memory card in
412 socket \fIS\fR where \fIS\fR represents a \fBPCMCIA\fR socket number.
413 .RE

415 .sp
416 .ne 2
417 .na
418 \fB/dev/rdisk/c\fIN\fRt\fIN\fRd\fIN\fRs\fIN\fR\fR
419 .ad
420 .RS 24n
421 Directory providing character device access for the \fBPCMCIA\fR memory card.
422 See OPERANDS for a description of \fIN\fR.
423 .RE

425 .sp
426 .ne 2
427 .na
428 \fB/dev/dsk/c\fIN\fRt\fIN\fRd\fIN\fRs\fIN\fR\fR
429 .ad
430 .RS 24n
431 Directory providing block device access for the \fBPCMCIA\fR memory card. See
432 OPERANDS for a description of \fIN\fR.
433 .RE

435 .SH SEE ALSO
436 .sp
437 .LP
438 \fBcpio\fR(1), \fBject\fR(1), \fBrmformat\fR(1), \fBtar\fR(1),
439 \fBvolcheck\fR(1), \fBmount\fR(1M), \fBnews\fR(1M), \fBprtvtoc\fR(1M),
440 \fBattributes\fR(5), \fBpcfs\fR(7FS)
441 \fBvolcheck\fR(1), \fBvolrmmount\fR(1), \fBmount\fR(1M), \fBnews\fR(1M),
442 \fBprtvtoc\fR(1M), \fBattributes\fR(5), \fBpcfs\fR(7FS)
441 .SS "x86 Only"
442 .sp
443 .LP
444 \fBfd\fR(7D)
445 .SH NOTES
446 .sp
447 .LP
448 A diskette or \fBPCMCIA\fR memory card containing a \fBufs\fR file system
449 created on a SPARC based system (by using \fBfdformat\fR and \fBnews\fR(1M)),
450 is not identical to a diskette or \fBPCMCIA\fR memory card containing a ufs
451 file system created on an x86 based system. Do not interchange ufs diskettes or
452 memory cards between these platforms. Use \fBcpio\fR(1) or \fBtar\fR(1) to

```

```

453 transfer files on diskettes or memory cards between them. A diskette or
454 \fBPCMCIA\fR memory card formatted using the \fB-t\fR \fBdos\fR option (or
455 \fB-d\fR) for \fBMS-DOS\fR does not have the necessary system files, and is
456 therefore not bootable. Trying to boot from it on a \fBPC\fR produces the
457 following message:
458 .sp
459 .in +2
460 .nf
461 Non-System disk or disk error.
462 Replace and strike any key when ready
463 .fi
464 .in -2
465 .sp

467 .SH BUGS
468 .sp
469 .LP
470 Currently, bad sector mapping is not supported on floppy diskettes or
471 \fBPCMCIA\fR memory cards. Therefore, a diskette or memory card is unusable if
472 \fBfdformat\fR finds an error (bad sector).

```

```

*****
14917 Tue Apr 22 07:25:50 2014
new/usr/src/man/man1/rmformat.1
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
*****
1 \" te
2 .\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved.
3 .\" Copyright 2014 Andrew Stormont.
4 #endif /* !codereview */
5 .\" The contents of this file are subject to the terms of the Common Development
6 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
7 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
8 .TH RMFORMAT 1 \"May 2, 2014\"
9 .TH RMFORMAT 1 \"Feb 19, 2009\"
10 .SH NAME
11 rmformat - removable rewritable media format utility
12 .SH SYNOPSIS
13 .LP
14 \fBrmformat\fR [\fB-DeHUV\fR] [\fB-b\fR \fIlabel\fR] [\fB-c\fR \fIblockno\fR]
15 [\fB-F\fRquick | long | force ] [\fB-s\fR \fIfilename\fR] [\fIdevname\fR]
16 .fi
18 .LP
19 .nf
20 \fBrmformat\fR \fB-V\fR read | write \fIdevname\fR
21 .fi
23 .LP
24 .nf
25 \fBrmformat\fR \fB-l\fR [\fIdevname\fR]
26 .fi
28 .SH DESCRIPTION
29 .sp
30 .LP
31 The \fBrmformat\fR utility is used to format, label, partition, and perform
32 other miscellaneous functions on removable, rewritable media that include
33 floppy drives, and the \fBPCMCIA\fR memory and \fBata\fR cards. The
34 \fBrmformat\fR utility should also be used with all USB mass storage devices,
35 including USB hard drives. This utility can also be used for the verification
36 and surface analysis and for repair of the bad sectors found during
37 verification if the drive or the driver supports bad block management.
38 .sp
39 .LP
40 After formatting, \fBrmformat\fR writes the label, which covers the full
41 capacity of the media as one slice on floppy and \fBPCMCIA\fR memory cards to
42 maintain compatibility with the behavior of \fBfdformat\fR. The partition
43 information can be changed with the help of other options provided by
44 \fBrmformat\fR.
45 .SH OPTIONS
46 .sp
47 .LP
48 The following options are supported:
49 .sp
50 .ne 2
51 .na
52 \fB-b\fR \fIlabel\fR
53 .ad
54 .sp .6
55 .RS 4n
56 Labels the media with a SUNOS label. A SUNOS volume label name is restricted to
57 8 characters. For media size greater than 1 TB, an EFI label is created. For
58 writing a \fBDOS\fR Volume label, the user should use \fBmkfs_pcfs\fR(1M).

```

```

59 .RE
61 .sp
62 .ne 2
63 .na
64 \fB-c\fR \fIblockno\fR
65 .ad
66 .sp .6
67 .RS 4n
68 Corrects and repairs the given block. This correct and repair option may not be
69 applicable to all devices supported by \fBformat\fR, as some devices may have
70 a drive with bad block management capability and others may have this option
71 implemented in the driver. If the drive or driver supports bad block
72 management, a best effort is made to rectify the bad block. If the bad block
73 still cannot be rectified, a message is displayed to indicate the failure to
74 repair. The block number can be provided in decimal, octal, or hexadecimal
75 format.
76 .sp
77 The normal floppy and \fBPCMCIA\fR memory and ata cards do not support bad
78 block management.
79 .RE
81 .sp
82 .ne 2
83 .na
84 \fB-D\fR
85 .ad
86 .sp .6
87 .RS 4n
88 Formats a 720KB (3.5 inch) double density diskette. This is the default for
89 double density type drives. This option is needed if the drive is a high or
90 extended-density type.
91 .RE
93 .sp
94 .ne 2
95 .na
96 \fB-e\fR
97 .ad
98 .sp .6
99 .RS 4n
100 Ejects the media upon completion. This feature may not be available if the
101 drive does not support motorized eject.
102 .RE
104 .sp
105 .ne 2
106 .na
107 \fB-F\fR quick | long | force
108 .ad
109 .sp .6
110 .RS 4n
111 Formats the media.
112 .sp
113 The \fBquick\fR option starts a format without certification or format with
114 limited certification of certain tracks on the media.
115 .sp
116 The \fBlong\fR option starts a complete format. For some devices this might
117 include the certification of the whole media by the drive itself.
118 .sp
119 The \fBforce\fR option to format is provided to start a long format without
120 user confirmation before the format is started.
121 .sp
122 In legacy media such as floppy drives, all options start a long format
123 depending on the mode (Extended Density mode, High Density mode, or Double
124 Density mode) with which the floppy drive operates by default. On \fBPCMCIA\fR

```

```

125 memory cards, all options start a long format.
126 .RE

128 .sp
129 .ne 2
130 .na
131 \fB\fB-H\fR\fR
132 .ad
133 .sp .6
134 .RS 4n
135 Formats a 1.44 MB (3.5 inch) high density diskette. This is the default for
136 high density type drives. It is needed if the drive is the Extended Density
137 type.
138 .RE

140 .sp
141 .ne 2
142 .na
143 \fB\fB-l\fR\fR
144 .ad
145 .sp .6
146 .RS 4n
147 Lists all removable devices. By default, without any options, \fBrmformat\fR
148 also lists all removable devices. If the \fBdev_name\fR is given,
149 \fBrmformat\fR lists the device associated with the \fBdev_name\fR. The output
150 shows the device pathname, vendor information, and the device type.
151 .RE

153 .sp
154 .ne 2
155 .na
156 \fB\fB-s\fR \fIfilename\fR\fR
157 .ad
158 .sp .6
159 .RS 4n
160 Enables the user to lay out the partition information in the SUNOS label.
161 .sp
162 The user should provide a file as input with information about each slice in a
163 format providing byte offset, size required, tags, and flags, as follows:
164 .sp
165 .in +2
166 .nf
167 slices: \fIn\fR = \fIoffset\fR, \fIsize\fR [, \fIflags\fR, \fItags\fR]
168 .fi
169 .in -2
170 .sp

172 where \fIn\fR is the slice number, \fIoffset\fR is the byte offset at which the
173 slice \fIn\fR starts, and \fIsize\fR is the required size for slice \fIn\fR.
174 Both \fIoffset\fR and \fIsize\fR must be a multiple of 512 bytes. These numbers
175 can be represented as decimal, hexadecimal, or octal numbers. No floating point
176 numbers are accepted. Details about maximum number of slices can be obtained
177 from the \fISystem Administration Guide: Basic Administration\fR.
178 .sp
179 To specify the \fIsize\fR or \fIoffset\fR in kilobytes, megabytes, or
180 gigabytes, add \fBKB\fR, \fBMB\fR, \fBGB\fR, respectively. A number without a
181 suffix is assumed to be a byte offset. The flags are represented as follows:
182 .sp
183 .in +2
184 .nf
185 \fBwm\fR = read-write, mountable
186 \fBwu\fR = read-write, unmountable
187 \fBwu\fR = read-only, unmountable
188 .fi
189 .in -2
190 .sp

```

```

192 The tags are represented as follows: \fBunassigned\fR, \fBboot\fR, \fBroot\fR,
193 \fBswap\fR, \fBusr\fR, \fBbackup\fR, \fBstand\fR, \fBvar\fR, \fBhome\fR,
194 \fBalternates\fR.
195 .sp
196 The tags and flags can be omitted from the four tuple when finer control on
197 those values is not required. It is required to omit both or include both. If
198 the tags and flags are omitted from the four tuple for a particular slice, a
199 default value for each is assumed. The default value for flags is \fBwm\fR and
200 for tags is \fBunassigned\fR.
201 .sp
202 Either full tag names can be provided or an abbreviation for the tags can be
203 used. The abbreviations can be the first two or more letters from the standard
204 tag names. \fBrmformat\fR is case insensitive in handling the defined tags &
205 flags.
206 .sp
207 Slice specifications are separated by :
208 .sp
209 For example:
210 .sp
211 .in +2
212 .nf
213 slices: 0 = 0, 30MB, "wm", "home" :
214           1 = 30MB, 51MB :
215           2 = 0, 100MB, "wm", "backup" :
216           6 = 81MB, 19MB
217 .fi
218 .in -2
219 .sp

221 \fBrmformat\fR does the necessary checking to detect any overlapping partitions
222 or illegal requests to addresses beyond the capacity of the media under
223 consideration. There can be only one slice information entry for each slice
224 \fIn\fR. If multiple slice information entries for the same slice \fIn\fR are
225 provided, an appropriate error message is displayed. The slice \fB2\fR is the
226 backup slice covering the whole disk capacity. The pound sign character,
227 \fB#\fR, can be used to describe a line of comments in the input file. If the
228 line starts with \fB#\fR, then \fBrmformat\fR ignores all the characters
229 following \fB#\fR until the end of the line.
230 .sp
231 Partitioning some of the media with very small capacity is permitted, but be
232 cautious in using this option on such devices.
233 .RE

235 .sp
236 .ne 2
237 .na
238 \fB\fB-U\fR\fR
239 .ad
240 .sp .6
241 .RS 4n
242 Performs \fBumount\fR on any file systems and then formats. See
243 \fBmount\fR(1M). This option unmounts all the mounted slices and issues a long
244 format on the device requested.
245 .RE

247 .sp
248 .ne 2
249 .na
250 \fB\fB-V\fR read | write\fR
251 .ad
252 .sp .6
253 .RS 4n
254 Verifies each block of media after format. The write verification is a
255 destructive mechanism. The user is queried for confirmation before the
256 verification is started. The output of this option is a list of block numbers,

```



```

257 which are identified as bad.
258 .sp
259 The read verification only verifies the blocks and report the blocks which are
260 prone to errors.
261 .sp
262 The list of block numbers displayed can be used with the \fB-c\fR option for
263 repairing.
264 .RE

266 .SH OPERANDS
267 .sp
268 .LP
269 The following operand is supported:
270 .sp
271 .ne 2
272 .na
273 \fB\fIdevname\fR\fR
274 .ad
275 .sp .6
276 .RS 4n
277 \fIdevname\fR can be provided as absolute device pathname or relative pathname
278 for the device from the current working directory or the nickname, such as
279 \fBcdrom\fR or \fBrdisk\fR.
280 .sp
281 For floppy devices, to access the first drive use \fB/dev/rdiskette0\fR (for
282 systems without volume management) or \fBfloppy0\fR (for systems with volume
283 management). Specify \fB/dev/rdiskette1\fR (for systems without volume
284 management) or \fBfloppy1\fR (for systems with volume management) to use the
285 second drive.
286 .sp
287 For systems without volume management running, the user can also provide the
288 absolute device pathname as \fB/dev/rdisk/c\fI? \fRt\fI? \fRd\fI? \fRs\fI? \fR\fR or
289 the appropriate relative device pathname from the current working directory.
290 .RE

292 .SH EXAMPLES
293 .LP
294 \fBExample 1 \fRFormatting a Diskette
295 .sp
296 .in +2
297 .nf
298 example$ \fBrmformat -F quick /dev/rdiskette\fR
299 Formatting will erase all the data on disk.
300 Do you want to continue? (y/n)\fBy\fR
301 .fi
302 .in -2
303 .sp

305 .LP
306 \fBExample 2 \fRFormatting a Diskette for a UFS File System
307 .sp
308 .LP
309 The following example formats a diskette and creates a UFS file system:

311 .sp
312 .in +2
313 .nf
314 example$ \fBrmformat -F quick /dev/aliases/floppy0\fR
315 Formatting will erase all the data on disk.
316 Do you want to continue? (y/n)\fBy\fR
317 example$ \fBsu\fR
318 # \fB/usr/sbin/newfs /dev/aliases/floppy0\fR
319 newfs: construct a new file system /dev/rdiskette: (y/n)? \fBy\fR
320 /dev/rdiskette: 2880 sectors in 80 cylinders of 2 tracks, 18 sectors
321      1.4MB in 5 cyl groups (16 c/g, 0.28MB/g, 128 i/g)
322 super-block backups (for fsck -F ufs -o b=#) at:

```

```

323 32, 640, 1184, 1792, 2336,
324 #
325 .fi
326 .in -2
327 .sp

329 .LP
330 \fBExample 3 \fRFormatting Removable Media for a PCFS File System
331 .sp
332 .LP
333 The following example shows how to create an alternate \fBfdisk\fR partition:

335 .sp
336 .in +2
337 .nf
338 example$ \fBrmformat -F quick /dev/rdisk/c0t4d0s2:c\fR
339 Formatting will erase all the data on disk.
340 Do you want to continue? (y/n)\fBy\fR
341 example$ \fBsu\fR
342 # \fBfdisk /dev/rdisk/c0t4d0s2:c\fR
343 # \fBmkfs -F pcfs /dev/rdisk/c0t4d0s2:c\fR
344 Construct a new FAT file system on /dev/rdisk/c0t4d0s2:c: (y/n)? \fBy\fR
345 #
346 .fi
347 .in -2
348 .sp

350 .sp
351 .LP
352 The following example describes how to create a \fBPCFS\fR file system
353 \fBwithout\fR an \fBfdisk\fR partition:

355 .sp
356 .in +2
357 .nf
358 example$ \fBrmformat -F quick /dev/rdiskette\fR
359 Formatting will erase all the data on disk.
360 Do you want to continue? (y/n)\fBy\fR
361 example$ \fBsu\fR
362 # \fBmkfs -F pcfs -o nofdisk,size=2 /dev/rdiskette\fR
363 Construct a new FAT file system on /dev/rdiskette: (y/n)? \fBy\fR
364 #
365 .fi
366 .in -2
367 .sp

369 .LP
370 \fBExample 4 \fRListing All Removable Devices
371 .sp
372 .LP
373 The following example shows how to list removable devices. This output shows a
374 long listing of such devices.

376 .sp
377 .in +2
378 .nf
379 example$ rmformat -l
380 Looking for devices...
381 Logical Node: /dev/rdisk/c5t0d0s2
382 Physical Node: /pci@1e,600000/usb@b/hub@2/storage@4/disk@0,0
383 Connected Device: TEAC FD-05PUB 1026
384 Device Type: Floppy drive
385 Bus: USB
386 Size: 1.4 MB
387 Label: floppy
388 Access permissions: Medium is not write protected.

```

```

389 .fi
390 .in -2
391 .sp

393 .SH FILES
394 .sp
395 .ne 2
396 .na
397 \fb\fb/dev/diskette0\fr\fr
398 .ad
399 .sp .6
400 .RS 4n
401 Directory providing block device access for the media in floppy drive 0.
402 .RE

404 .sp
405 .ne 2
406 .na
407 \fb\fb/dev/rdiskette0\fr\fr
408 .ad
409 .sp .6
410 .RS 4n
411 Directory providing character device access for the media in floppy drive 0.
412 .RE

414 .sp
415 .ne 2
416 .na
417 \fb\fb/dev/aliases\fr\fr
418 .ad
419 .sp .6
420 .RS 4n
421 Directory providing symbolic links to the character devices for the different
422 media under the control of volume management using appropriate alias.
423 .RE

425 .sp
426 .ne 2
427 .na
428 \fb\fb/dev/aliases/floppy0\fr\fr
429 .ad
430 .sp .6
431 .RS 4n
432 Symbolic link to the character device for the media in floppy drive 0.
433 .RE

435 .sp
436 .ne 2
437 .na
438 \fb\fb/dev/rdiskette\fr\fr
439 .ad
440 .sp .6
441 .RS 4n
442 Symbolic link providing character device access for the media in the primary
443 floppy drive, usually drive 0.
444 .RE

446 .sp
447 .ne 2
448 .na
449 \fb\fb/dev/dsk\fr\fr
450 .ad
451 .sp .6
452 .RS 4n
453 Directory providing block device access for the \fbPCMCIA\fr memory and ata
454 cards and removable media devices.

```

```

455 .RE

457 .sp
458 .ne 2
459 .na
460 \fb\fb/dev/rdsk\fr\fr
461 .ad
462 .sp .6
463 .RS 4n
464 Directory providing character device access for the \fbPCMCIA\fr memory and ata
465 cards and removable media devices.
466 .RE

468 .sp
469 .ne 2
470 .na
471 \fb\fb/dev/aliases/pcmemS\fr\fr
472 .ad
473 .sp .6
474 .RS 4n
475 Symbolic link to the character device for the \fbPCMCIA\fr memory card in
476 socket S, where S represents a \fbPCMCIA\fr socket number.
477 .RE

479 .sp
480 .ne 2
481 .na
482 \fb\fb/dev/aliases/rmdisk0\fr\fr
483 .ad
484 .sp .6
485 .RS 4n
486 Symbolic link to the generic removable media device that is not a \fbCD-ROM\fr,
487 floppy, \fbDVD-ROM\fr, \fbPCMCIA\fr memory card, and so forth.
488 .RE

490 .sp
491 .ne 2
492 .na
493 \fb\fb/dev/rdsk\fr\fr
494 .ad
495 .sp .6
496 .RS 4n
497 Directory providing character device access for the \fbPCMCIA\fr memory and ata
498 cards and other removable devices.
499 .RE

501 .sp
502 .ne 2
503 .na
504 \fb\fb/dev/dsk\fr\fr
505 .ad
506 .sp .6
507 .RS 4n
508 Directory providing block device access for the \fbPCMCIA\fr memory and ata
509 cards and other removable media devices.
510 .RE

512 .SH SEE ALSO
513 .sp
514 .LP
515 \fbCpio\fr(1), \fbEject\fr(1), \fbFdformat\fr(1), \fbTar\fr(1),
516 \fbVlcheck\fr(1), \fbFormat\fr(1M), \fbMkfs_pcfs\fr(1M), \fbMount\fr(1M),
517 \fbNewfs\fr(1M), \fbPrvtoc\fr(1M), \fbRmount\fr(1M), \fbRpc.smsrverd\fr(1M),
518 \fbAttributes\fr(5), \fbScsausb\fr(7D), \fbSd\fr(7D), \fbPcfs\fr(7FS),
519 \fbUdfs\fr(7FS)
520 \fbVlcheck\fr(1), \fbVlrmount\fr(1), \fbFormat\fr(1M), \fbMkfs_pcfs\fr(1M),

```

```
512 \fBmount\fR(1M), \fBnewfs\fR(1M), \fBprtvtoc\fR(1M), \fBbrmount\fR(1M),
513 \fBbrpc.smsrverd\fR(1M), \fBattributes\fR(5), \fBscsa2usb\fR(7D), \fBsd\fR(7D),
514 \fBpcfs\fR(7FS), \fBudfs\fR(7FS)
520 .sp
521 .LP
522 \fISystem Administration Guide: Basic Administration\fR
523 .SH NOTES
524 .sp
525 .LP
526 A rewritable media or \fBPCMCIA\fR memory card or \fBPCMCIA\fR ata card
527 containing a \fBufs\fR file system created on a SPARC-based system (using
528 \fBnewfs\fR(1M)) is not identical to a rewritable media or \fBPCMCIA\fR memory
529 card containing a \fBufs\fR file system created on an x86 based system. Do not
530 interchange any removable media containing \fBufs\fR between these platforms;
531 use \fBcpio\fR(1) or \fBtar\fR(1) to transfer files on diskettes or memory
532 cards between them. For interchangeable filesystems refer to \fBpcfs\fR(7FS)
533 and \fBudfs\fR(7FS).
534 .sp
535 .LP
536 \fBrmformat\fR might not list all removable devices in virtualization
537 environments.
538 .SH BUGS
539 .sp
540 .LP
541 Currently, bad sector mapping is not supported on floppy diskettes or
542 \fBPCMCIA\fR memory cards. Therefore, a diskette or memory card is unusable if
543 \fBrmformat\fR finds an error (\fBbad sector\fR).
```

new/usr/src/man/man1m/rmmount.1m

1

```
*****
5890 Tue Apr 22 07:25:50 2014
new/usr/src/man/man1m/rmmount.1m
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
*****
1 \" te
2 .\" Copyright (c) 2004, Sun Microsystems, Inc. All Rights Reserved
3 .\" Copyright 2014 Andrew Stormont.
4 #endif /* !codereview */
5 .\" The contents of this file are subject to the terms of the Common Development
6 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
7 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
8 .TH RMMOUNT 1M \"May 2, 2014\"
9 .TH RMMOUNT 1M \"Mar 1, 2007\"
10 .SH NAME
11 rmmount \- removable media mouter for CD-ROM, floppy, Jaz drive, and others
12 .SH SYNOPSIS
13 .LP
14 \fB/usr/sbin/rmmount\fR [\fB-D\fR]
15 .fi
17 .SH DESCRIPTION
18 .sp
19 .LP
20 The \fBrmmount\fR utility is a removable media mouter that is executed by
21 volume management whenever a removable medium, such as a \fBCD-ROM\fR or a
22 floppy, is inserted. Removable media is managed by an application or a volume
23 manager.
24 manager. \fBrmmount\fR can also be called by using \fBvolrmmount\fR(1).
25 .sp
26 Upon insertion of a medium and following invocation of the \fBvolcheck\fR(1)
27 command, \fBrmmount\fR determines what type of file system (if any) is on that
28 medium. If a file system is present, \fBrmmount\fR mounts the file system in
29 one of the locations listed below.
30 .sp
31 .LP
32 For a diskette (floppy):
33 .sp
34 .ne 2
35 .na
36 \fB\fB/floppy/floppy0\fR\fR
37 .ad
38 .RS 26n
39 symbolic link to mounted floppy in local floppy drive
40 .RE
42 .sp
43 .ne 2
44 .na
45 \fB\fB/floppy/floppy_name\fR\fR
46 .ad
47 .RS 26n
48 mounted named floppy
49 .RE
51 .sp
52 .ne 2
53 .na
54 \fB\fB/floppy/unnamed_floppy\fR\fR
55 .ad
56 .RS 26n
57 mounted unnamed floppy
```

new/usr/src/man/man1m/rmmount.1m

2

```
58 .RE
60 .sp
61 .LP
62 For a CD-ROM or a DVD-ROM:
63 .sp
64 .ne 2
65 .na
66 \fB\fB/cdrom/cdrom0\fR\fR
67 .ad
68 .sp .6
69 .RS 4n
70 symbolic link to mounted \fBCD-ROM\fR in local \fBCD-ROM\fR drive
71 .RE
73 .sp
74 .ne 2
75 .na
76 \fB\fB/cdrom/CD-ROM_name\fR\fR
77 .ad
78 .sp .6
79 .RS 4n
80 mounted named \fBCD-ROM\fR
81 .RE
83 .sp
84 .ne 2
85 .na
86 \fB\fB/cdrom/CD-ROM_name/partition\fR\fR
87 .ad
88 .sp .6
89 .RS 4n
90 mounted named \fBCD-ROM\fR with partitioned file system
91 .RE
93 .sp
94 .ne 2
95 .na
96 \fB\fB/cdrom/unnamed_cdrom\fR\fR
97 .ad
98 .sp .6
99 .RS 4n
100 mounted unnamed \fBCD-ROM\fR
101 .RE
103 .sp
104 .LP
105 For a Zip drive:
106 .sp
107 .ne 2
108 .na
109 \fB\fB/rmdisk/zip0\fR\fR
110 .ad
111 .RS 30n
112 symbolic link to mounted Zip medium in local Zip drive
113 .RE
115 .sp
116 .ne 2
117 .na
118 \fB\fB/rmdisk/\fIZip_name\fR\fR\fR
119 .ad
120 .RS 30n
121 mounted named Zip medium
122 .RE
```

```

124 .sp
125 .ne 2
126 .na
127 \fB\fB/rmdisk/\fIZip_name\fR/partition\fR\fR
128 .ad
129 .RS 30n
130 mounted named Zip medium with partitioned file system
131 .RE

133 .sp
134 .ne 2
135 .na
136 \fB\fB/rmdisk/unnamed_zip\fR\fR
137 .ad
138 .RS 30n
139 mounted unnamed Zip medium
140 .RE

142 .sp
143 .LP
144 For a Jaz drive:
145 .sp
146 .ne 2
147 .na
148 \fB\fB/rmdisk/jaz0\fR\fR
149 .ad
150 .RS 30n
151 symbolic link to mounted Jaz medium in local Jaz drive
152 .RE

154 .sp
155 .ne 2
156 .na
157 \fB\fB/rmdisk/\fIJaz_name\fR\fR\fR
158 .ad
159 .RS 30n
160 mounted named Jaz medium
161 .RE

163 .sp
164 .ne 2
165 .na
166 \fB\fB/rmdisk/\fIJaz_name\fR/partition\fR\fR
167 .ad
168 .RS 30n
169 mounted named Jaz medium with partitioned file system
170 .RE

172 .sp
173 .ne 2
174 .na
175 \fB\fB/rmdisk/unnamed_Jaz\fR\fR
176 .ad
177 .RS 30n
178 mounted unnamed Jaz medium
179 .RE

181 .sp
182 .LP
183 For a generic "rmdisk" drive:
184 .sp
185 .ne 2
186 .na
187 \fB\fB/rmdisk/rmdisk0\fR\fR
188 .ad
189 .sp .6

```

```

190 .RS 4n
191 symbolic link to mounted removable medium in local removable medium drive
192 .RE

194 .sp
195 .ne 2
196 .na
197 \fB\fB/rmdisk/\fIrmdisk_name\fR\fR\fR
198 .ad
199 .sp .6
200 .RS 4n
201 mounted named removable medium
202 .RE

204 .sp
205 .ne 2
206 .na
207 \fB\fB/rmdisk/\fIrmdisk_name\fR/partition\fR\fR
208 .ad
209 .sp .6
210 .RS 4n
211 mounted named removable medium with partitioned file system
212 .RE

214 .sp
215 .ne 2
216 .na
217 \fB\fB/rmdisk/unnamed_rmdisk\fR\fR
218 .ad
219 .sp .6
220 .RS 4n
221 mounted unnamed removable medium
222 .RE

224 .sp
225 .LP
226 If the media is read-only (for example, a \fBCD-ROM\fR or a floppy with
227 write-protect tab set), the file system is mounted read-only.
228 .sp
229 .LP
230 If a file system is not identified, \fB\rmount\fR does not mount a file system.
231 See the \fI\fR for more information on the location of \fBCD-ROM\fR, floppy,
232 and other media without file systems.
233 .sp
234 .LP
235 If a file system type has been determined, it is then checked to see that it is
236 "clean." If the file system is "dirty," \fBfsck\fR \fB-p\fR (see
237 \fBfsck\fR(1M)) is run in an attempt to clean it. If \fBfsck\fR fails, the file
238 system is mounted read-only.
239 .sp
240 .LP
241 After the mount is complete, "actions" associated with the media type are
242 executed. These actions allow for the notification to other programs that new
243 media are available.
244 .sp
245 .LP
246 Actions are executed in the order in which they appear in the configuration
247 file. The action function can return either \fB1\fR or \fB0\fR. If it returns
248 \fB0\fR, no further actions will be executed. This allows the function to
249 control which applications are executed.
250 .sp
251 .LP
252 In order to execute an action, \fB\rmount\fR performs a \fBdlopen\fR(3C) on the
253 shared object and calls the action function defined within it. The definition
254 of the interface to actions can be found in \fB/usr/include/rmmount.h\fR.
255 .sp

```

```
256 .LP
257 File systems mounted by \fBrmmount\fR are always mounted with the \fBnosuid\fR
258 flag set, thereby disabling setuid programs and access to block or character
259 devices in that file system. Upon ejection, \fBrmmount\fR unmounts mounted file
260 systems and executes actions associated with the media type. If a file system
261 is "busy" (that is, it contains the current working directory of a live
262 process), the ejection will fail.
263 .SH OPTIONS
264 .sp
265 .ne 2
266 .na
267 \fB\fB-D\fR\fR
268 .ad
269 .RS 6n
270 Turn on the debugging output from the \fBrmmount\fR \fBdprintf\fR calls.
271 .RE

273 .SH FILES
274 .sp
275 .ne 2
276 .na
277 \fB\fB/usr/lib/rmmount/*.so.1\fR\fR
278 .ad
279 .RS 27n
280 shared objects used by \fBrmmount\fR.
281 .RE

283 .SH SEE ALSO
284 .sp
285 .LP
286 \fB\fBvolcheck\fR(1), \fB\fBfsck\fR(1M), \fB\fBdlopen\fR(3C),
281 \fB\fBvolcheck\fR(1), \fB\fBvolrmmount\fR(1), \fB\fBfsck\fR(1M), \fB\fBdlopen\fR(3C),
287 \fB\fBattributes\fR(5)
288 .sp
289 .LP
290 \fI\fR
```

new/usr/src/pkg/manifests/service-storage-media-volume-manager.mf 1

```
*****
2802 Tue Apr 22 07:25:50 2014
new/usr/src/pkg/manifests/service-storage-media-volume-manager.mf
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright 2012 Nexenta Systems, Inc. All rights reserved.
25 # Copyright 2014 Andrew Stormont.
26 #endif /* ! codereview */
27 #
28 #
29 set name=pkg.fmri value=pkg:/service/storage/media-volume-manager@$(PKGVERS)
30 set name=pkg.description value="Non-graphical removable volume manager"
31 set name=pkg.summary value="Removable volume manager"
32 set name=info.classification value=org.opensolaris.category.2008:System/Media
33 set name=variant.arch value=$(ARCH)
34 dir path=lib/variant.opensolaris.zone=global
35 dir path=lib/svc/variant.opensolaris.zone=global
36 dir path=lib/svc/manifest/group=sys/variant.opensolaris.zone=global
37 dir path=lib/svc/manifest/system/group=sys/variant.opensolaris.zone=global
38 dir path=lib/svc/manifest/system/filesystem/group=sys \
39   variant.opensolaris.zone=global
40 dir path=lib/svc/method/variant.opensolaris.zone=global
41 dir path=usr/group=sys
42 dir path=usr/bin
43 dir path=usr/lib
44 dir path=usr/sbin
45 dir path=usr/share/man/man1
46 dir path=usr/share/man/man1m
47 file path=lib/svc/manifest/system/filesystem/rmvolmgr.xml group=sys mode=0444 \
48   variant.opensolaris.zone=global
49 file path=lib/svc/method/svc-rmvolmgr mode=0555 \
50   variant.opensolaris.zone=global
51 file path=usr/bin/rmformat mode=4555
52 file path=usr/bin/rmmount mode=0555
53 file path=usr/bin/volcheck mode=0555
25 file path=usr/bin/volrmmount mode=0555
54 file path=usr/lib/rmvolmgr mode=0555
55 file path=usr/share/man/man1/rmformat.1
56 file path=usr/share/man/man1/rmmount.1
57 file path=usr/share/man/man1/volcheck.1
30 file path=usr/share/man/man1/volrmmount.1
```

new/usr/src/pkg/manifests/service-storage-media-volume-manager.mf 2

```
58 file path=usr/share/man/man1m/rmmount.1m
59 file path=usr/share/man/man1m/rmvolmgr.1m
60 legacy pkg=SUNWrmvolmgr desc="Non-graphical removable volume manager" \
61   name="Removable volume manager"
62 legacy pkg=SUNWrmvolmgr desc="Non-graphical removable volume manager (Root)" \
63   name="Removable volume manager (Root)"
64 license cr_Sun license=cr_Sun
65 license lic_CDDL license=lic_CDDL
66 link path=usr/bin/rmumount target=./rmmount
67 link path=usr/sbin/rmmount target=./bin/rmmount
68 link path=usr/share/man/man1/rmumount.1 target=rmmount.1
```