```
**********************************************************
   10877 Tue May 13 22:52:05 2014
new/usr/src/cmd/Makefile
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

# Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2010 Nexenta Systems, Inc.  All rights reserved.
# Copyright (c) 2012 Joyent, Inc.  All rights reserved.
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright (c) 2013 DEY Storage Systems, Inc. All rights reserved.
# Copyright 2014 Andrew Stormont.
#endif /* ! codereview */

include ../Makefile.master

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

FIRST_SUBDIRS=          \
        isaexec         \
        platexec

COMMON_SUBDIRS=         \
        allocate        \
        availdevs       \
        lp              \
        perl            \
        man             \
        Adm             \
        abi             \
```



```
        adbgen          \
        acct            \
        acctadm         \
        arch            \
        asa             \
        ast             \
        audio           \
        auths           \
        autopush        \
        avs             \
        awk             \
        awk_xpg4        \
        backup          \
        banner          \
        bart            \
        basename        \
        bc              \
        bdiff           \
        beadm           \
        bfs             \
        bnu             \
        boot            \
        busstat         \
        cal             \
        calendar        \
        captoinfo       \
        cat             \
        cdrw            \
        cfgadm          \
        checkeq         \
        checknr         \
        chgrp           \
        chmod           \
        chown           \
        chroot          \
        clear           \
        clinfo          \
        cmd-crypto      \
        cmd-inet        \
        col             \
        compress        \
        consadm         \
        coreadm         \
        cpio            \
        cpc             \
        cron            \
        crypt           \
        csh             \
        csplit          \
        ctrun           \
        ctstat          \
        ctwatch         \
        datadm          \
        date            \
        dc              \
        dd              \
        deroff          \
        devfsadm        \
        syseventd       \
        devctl          \
        devinfo         \
        devmgmt         \
        devprop         \
        dfs.cmds        \
        diff            \
        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         eeprom          \
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         dlmgmtd         \
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         mesg            \
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         prtvtoc         \
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         sdpadm          \
366         sed             \
367         sendmail        \
368         setfacl         \
369         setmnt          \
370         setpgrp         \
371         setuname        \
372         sgs             \
373         sh              \
374         shcomp          \
375         smbios          \
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         vrrpadm         \
442         vscan           \
443         vt              \
444         w               \
445         wall            \
446         which           \
447         who             \
448         whodo           \
449         wracct          \
450         write           \
451         wusbadm         \
452         xargs           \
453         xstr            \
454         yes             \
```



```
455         ypcmd           \
456         yppasswd        \
457         zdb             \
458         zdump           \
459         zfs             \
460         zhack           \
461         zic             \
462         zinject         \
463         zlogin          \
464         zoneadm         \
465         zoneadmd        \
466         zonecfg         \
467         zonename        \
468         zpool           \
469         zlook           \
470         zonestat        \
471         zstreamdump     \
472         ztest

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

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

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        \
```

```
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          \
```



```
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         mesg            \
623         mkdir           \
624         mkpwdict        \
625         mktemp          \
626         more            \
627         mpathadm        \
628         msgfmt          \
629         mv              \
630         ndmpadm         \
631         ndmpstat        \
632         newgrp          \
633         newtask         \
634         nice            \
635         nohup           \
636         oawk            \
637         pack            \
638         passwd          \
639         passmgmt        \
640         pathchk         \
641         pfexec          \
642         pg              \
643         pgrep           \
644         picl            \
645         pools           \
646         power           \
647         pr              \
648         praudit         \
649         print           \
650         profiles        \
651         projadd         \
652         projects        \
```

```
653         prstat          \
654         prtdiag         \
655         ps              \
656         psrinfo         \
657         ptools          \
658         pwconv          \
659         pwd             \
660         pyzfs           \
661         raidctl         \
662         ramdiskadm      \
663         rcap            \
664         rcm_daemon      \
665         refer           \
666         regcmp          \
667         renice          \
668         roles           \
669         rm              \
670         rmdir           \
671         rmformat        \
672         rmmount         \
673         rmvolmgr        \
674         sasinfo         \
675         sbdadm          \
676         scadm           \
677         script          \
678         scsi            \
679         sdiff           \
680         sdpadm          \
681         sgs             \
682         sh              \
683         shcomp          \
684         smbsrv          \
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         zoneadmd        \
737         zonecfg         \
738         zonename        \
739         zpool           \
740         zonestat

742 sparc_MSGSUBDIRS=        \
743         fruadm          \
744         prtdscp         \
745         prtfru          \
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:
```

```
**********************************************************
    1589 Tue May 13 22:52:05 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>
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License, Version 1.0 only
# (the "License").  You may not use this file except in compliance
# with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Copyright 2014 Andrew Stormont.
#
#endif /* ! codereview */

PROG= rmformat

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

include ../Makefile.cmd

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

LDLIBS +=       -lsmedia -lvolmgt -ladm -lefi

CERRWARN += -_gcc=-Wno-uninitialized

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

$(ROOTBIN)/rmformat := FILEMODE = 04555

.KEEP_STATE:

all: $(PROG)

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

install: all $(ROOTPROG)

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



```
lint: lint_SRCS

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

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

include ../Makefile.targ
```

```
**********************************************************
   30011 Tue May 13 22:52:05 2014
new/usr/src/cmd/rmformat/rmf_menu.c
4833 Remove volrmmount
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 *
 *
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
 */

/*
 * rmf_menu.c :
 *      Command line options to rmformat are processed in this file.
 */

#include "rmformat.h"
#include <sys/smedia.h>
#include <priv_utils.h>

extern int32_t D_flag;
extern int32_t e_flag;
extern int32_t H_flag;
extern int32_t U_flag;
extern int32_t V_flag;
extern int32_t b_flag;
extern int32_t w_flag;
extern int32_t W_flag;
extern int32_t s_flag;
extern int32_t c_flag;
extern int32_t F_flag;
extern int32_t R_flag;
extern int32_t p_flag;
extern int32_t l_flag;

extern char *myname;
extern char *slice_file;
extern diskaddr_t repair_blk_no;
extern int32_t quick_format;
extern int32_t long_format;
```



```
extern int32_t force_format;
extern int32_t rw_protect_enable;
extern int32_t rw_protect_disable;
extern int32_t wp_enable_passwd;
extern int32_t wp_disable_passwd;
extern int32_t wp_enable;
extern int32_t wp_disable;
extern int32_t verify_write;
extern char *dev_name;
extern char *label;
extern int total_devices_found;
extern int removable_found;
char *global_intr_msg;
smmedium_prop_t med_info;
int vol_running;

extern void check_invalid_combinations();
extern void check_invalid_combinations_again(int32_t);
extern void process_options();
extern void get_passwd(struct smwp_state *wp, int32_t confirm);
extern int32_t valid_slice_file(smedia_handle_t, int32_t, char *,
        struct extvtoc *);
extern void trap_SIGINT();
extern void release_SIGINT();
extern int32_t verify(smedia_handle_t handle, int32_t fd,
        diskaddr_t start_sector, uint32_t nblocks,
        char *buf, int32_t flag, int32_t blocksize, int32_t no_raw_rw);
extern void my_perror(char *err_string);
extern void write_default_label(smedia_handle_t, int32_t fd);
extern int find_device(int defer, char *tmpstr);

void overwrite_metadata(int32_t fd, smedia_handle_t handle);

int32_t write_sunos_label(int32_t fd, int32_t media_type);

int32_t my_open(char *device_name, int32_t flags);

int32_t check_and_unmount_vold(char *device_name, int32_t flag);
int32_t check_and_unmount_scsi(char *device_name, int32_t flag);

int32_t check_and_unmount_floppy(int32_t fd, int32_t flag);
int32_t get_confirmation(void);


static void     process_F_flag(smedia_handle_t handle, int32_t fd);
static void     process_w_flag(smedia_handle_t handle);
static void     process_W_flag(smedia_handle_t handle);
static void     process_R_flag(smedia_handle_t handle);
void            process_p_flag(smedia_handle_t handle, int32_t fd);
static void     process_c_flag(smedia_handle_t handle);
static void     process_V_flag(smedia_handle_t handle, int32_t fd);
static void     process_s_flag(smedia_handle_t, int32_t fd);
static void     process_e_flag(smedia_handle_t handle);
static void     process_H_flag(smedia_handle_t handle, int32_t fd);
static void     process_D_flag(smedia_handle_t handle, int32_t fd);
static void     process_b_flag(int32_t fd);
static void     process_l_flag(void);

void
process_options()
{
        int32_t fd;
        smedia_handle_t handle;
        int32_t m_scsi_umount = 0;
        int32_t m_flp_umount = 0;
        int32_t v_device_umount = 0;
```

```
121         int32_t umount_required = 0;
122         int32_t removable;
123         int32_t umount_failed = 0;
124         struct dk_minfo media;

126         check_invalid_combinations();

128         if (l_flag && !dev_name) {
129                 process_l_flag();
130                 return;
131         }

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

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

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

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

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

171         /* Check if volume manager has mounted this */
172         if (umount_required) {
173                 m_scsi_umount = check_and_unmount_scsi(dev_name, U_flag);
105                 v_device_umount = check_and_unmount_vold(dev_name, U_flag);
106                 if (v_device_umount != 1) {
107                         m_scsi_umount = check_and_unmount_scsi(dev_name,
108                             U_flag);
174                 if (m_scsi_umount != 1) {
175                         m_flp_umount = check_and_unmount_floppy(fd, U_flag);
110                                 m_flp_umount = check_and_unmount_floppy(fd,
111                                     U_flag);
176                         if (m_flp_umount != 1) {
177                                 umount_failed = 1;
178                         }
179                 }
180         }
```



```
117         }

182         if (umount_required && U_flag && umount_failed) {
183                 if (m_scsi_umount || m_flp_umount) {
120                 if (v_device_umount || m_scsi_umount || m_flp_umount) {
184                         (void) fprintf(stderr,
185                             gettext("Could not unmount device.\n"));
186                         (void) close(fd);
187                         exit(1);
188                 }
189         }

191         if (umount_required && !U_flag) {
192                 if (m_scsi_umount || m_flp_umount) {
129                 if (v_device_umount || m_scsi_umount || m_flp_umount) {
193                         (void) fprintf(stderr, gettext("Device mounted.\n"));
194                         (void) fprintf(stderr,
195                             gettext("Requested operation can not be \
196 performed on a mounted device.\n"));
197                         (void) close(fd);
198                         exit(1);
199                 }
200         }
201         /* register the fd with the libsmedia */
202         handle = smedia_get_handle(fd);
203         if (handle == NULL) {
204                 (void) fprintf(stderr,
205                     gettext("Failed to get libsmedia handle.\n"));
206                 (void) close(fd);
207                 exit(1);
208         }

210         if (smedia_get_medium_property(handle, &med_info) < 0) {
211                 (void) fprintf(stderr,
212                     gettext("Get medium property failed \n"));
213                 (void) smedia_release_handle(handle);
214                 (void) close(fd);
215                 exit(1);
216         }

218         DPRINTF1("media type %x\n", med_info.sm_media_type);
219         DPRINTF1("media block size %x\n", med_info.sm_blocksize);
220         DPRINTF1("media capacity %u\n", (uint32_t)med_info.sm_capacity);
221         DPRINTF3("media cyl %d head %d sect %d\n",
222             med_info.sm_pcyl, med_info.sm_nhead, med_info.sm_nsect);
223         check_invalid_combinations_again(med_info.sm_media_type);

225         /*
226          * Special handling for pcmcia, sometimes open the file in
227          * read-write mode.
228          */

230         if (med_info.sm_media_type == SM_PCMCIA_MEM) {
231                 if (F_flag || H_flag || D_flag || (V_flag && verify_write)) {
232                         (void) close(fd);
233                         DPRINTF("Reopening device\n");
234                         fd = my_open(dev_name, O_RDWR|O_NDELAY);
235                         if (fd < 0)  {
236                                 PERROR("Could not open device");
237                                 (void) smedia_release_handle(handle);
238                                 (void) close(fd);
239                                 exit(1);
240                         }
241                 }
242         }
```

```
244         if (med_info.sm_media_type == SM_PCMCIA_ATA) {
245                 if (V_flag || c_flag) {
246                         (void) fprintf(stderr,
247                             gettext("Option not supported on PC ATA cards\n"));
248                         (void) smedia_release_handle(handle);
249                         (void) close(fd);
250                         exit(1);
251                 }
252                 if (F_flag) {
253                         /* same text as used by the format command */
254                         (void) fprintf(stderr,
255                             gettext("Cannot format this drive. Please use your \
256 Manufacturer supplied formatting utility.\n"));
257                         (void) smedia_release_handle(handle);
258                         (void) close(fd);
259                         exit(1);
260                 }
261         }

263         if (F_flag)
264                 process_F_flag(handle, fd);
265         if (w_flag)
266                 process_w_flag(handle);
267         if (W_flag)
268                 process_W_flag(handle);
269         if (R_flag)
270                 process_R_flag(handle);
271         if (p_flag)
272                 process_p_flag(handle, fd);
273         if (D_flag)
274                 process_D_flag(handle, fd);
275         if (H_flag)
276                 process_H_flag(handle, fd);
277         if (V_flag)
278                 process_V_flag(handle, fd);
279         if (c_flag)
280                 process_c_flag(handle);
281         if (b_flag)
282                 process_b_flag(fd);
283         if (s_flag)
284                 process_s_flag(handle, fd);
285         if (e_flag)
286                 process_e_flag(handle);
287         if (l_flag) {
288                 process_l_flag();
289         }

291         (void) smedia_release_handle(handle);
292         (void) close(fd);
293 }

295 /*
296  * This routine handles the F_flag.
297  * This options should not be used for floppy. However,
298  * if this option is used for floppy, the option will
299  * be forced to SM_FORMAT_HD and smedia_format is called.
300  * Note that smedia_format is a blocked mode format and it
301  * returns only after the complete formatting is over.
302  */

304 static void
305 process_F_flag(smedia_handle_t handle, int32_t fd)
306 {
307         uint32_t format_flag = 0;
244         uint32_t format_flag;
308         int32_t old_per = 0;
```



```
309         int32_t new_per, ret_val;

311         if (force_format) {
312                 (void) fprintf(stderr,
313                     gettext("Formatting disk.\n"));
314         } else {
315                 (void) fprintf(stderr,
316                     gettext("Formatting will erase all the data on disk.\n"));
317                 if (!get_confirmation())
318                         return;
319         }

321         if (quick_format)
322                 format_flag = SM_FORMAT_QUICK;
323         else if (long_format)
324                 format_flag = SM_FORMAT_LONG;
325         else if (force_format)
326                 format_flag = SM_FORMAT_FORCE;

328         if (med_info.sm_media_type == SM_FLOPPY)
329                 format_flag = SM_FORMAT_HD;

331         if ((med_info.sm_media_type != SM_FLOPPY) &&
332             (med_info.sm_media_type != SM_PCMCIA_MEM) &&
333             (med_info.sm_media_type != SM_SCSI_FLOPPY)) {
334                 global_intr_msg = "Interrupting format may render the \
335 medium useless";
336         } else {
337                 global_intr_msg = "";
338         }
339                 trap_SIGINT();

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

356         /* CONSTCOND */
357         while (1) {
358                 ret_val = smedia_check_format_status(handle);
359                 if (ret_val == -1) {
360                         if (errno != ENOTSUP) {
361                                 PERROR("Format failed");
362                                 (void) smedia_release_handle(handle);
363                                 (void) close(fd);
364                                 exit(1);
365                         } else {
366                                 /* Background formatting is not supported */
367                                 break;
368                         }
369                 }
370                 if (ret_val == 100) {
371                         (void) printf("\n");
372                         (void) fflush(stdout);
373                         break;
374                 }
```

```
375                 new_per = (ret_val * 80)/100;
376                 while (new_per >= old_per) {
377                         (void) printf(".");
378                         (void) fflush(stdout);
379                         old_per++;
380                 }
381                 (void) sleep(6);
382         }

384         if ((med_info.sm_media_type == SM_FLOPPY) ||
385             (med_info.sm_media_type == SM_PCMCIA_MEM) ||
386             (med_info.sm_media_type == SM_SCSI_FLOPPY)) {
387                 (void) write_sunos_label(fd, med_info.sm_media_type);
388         } else {

390                 /*
391                  * Iomega drives don't destroy the data in quick format.
392                  * Do a best effort write to first 1024 sectors.
393                  */

395                 if (quick_format)
396                         overwrite_metadata(fd, handle);

398                 (void) write_default_label(handle, fd);
399         }

401         release_SIGINT();
402 }

404 /*
405  * List removable devices.
406  */
407 static void
408 process_l_flag()
409 {
410         int retry;
411         int removable;
412         int total_devices_found_last_time;
413         int defer = 0;
414         char *tmpstr = NULL;
351         char *tmpstr;

416 #define MAX_RETRIES_FOR_SCANNING 3

418         vol_running = volmgt_running();
419         if (vol_running)
420                 defer = 1;
421         (void) printf(gettext("Looking for devices...\n"));
422         total_devices_found_last_time = 0;

424         /*
425          * Strip out any leading path.  For example, /dev/rdsk/c3t0d0s2
426          * will result in tmpstr = c3t0d0s2.  dev_name is given as input
427          * argument.
428          */
429         if (dev_name) {
430                 if ((tmpstr = strrchr(dev_name, '/')) != NULL) {
431                         tmpstr += sizeof (char);
432                 } else {
433                         tmpstr = dev_name;
434                 }
435         }

437         for (retry = 0; retry < MAX_RETRIES_FOR_SCANNING; retry++) {
438                 removable = find_device(defer, tmpstr);
439                 if (removable == -1)
```



```
440                         break;

442                 /*
443                  * We'll do a small sleep and retry the command if volume
444                  * manager is running and no removable devices are found.
445                  * This is because the device may be busy.
446                  */
447                 if (defer || (vol_running && (removable == 0))) {
448                         if ((total_devices_found == 0) ||
449                             (total_devices_found !=
450                             total_devices_found_last_time)) {
451                                 total_devices_found_last_time =
452                                     total_devices_found;
453                                 (void) sleep(2);
454                         } else {
455                                 /* Do the printing this time */
456                                 defer = 0;
457                                 removable_found = 0;
458                         }

460                 } else
461                         break;
462         }
463         if (removable_found == 0)
464                 (void) printf(gettext("No removables found.\n"));
465 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   43460 Tue May 13 22:52:05 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>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 *
 *
 */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
 */

/*
 * rmf_misc.c :
 *      Miscelleneous routines for rmformat.
 */

#include <sys/types.h>
#include <stdio.h>
#include <sys/mnttab.h>
#include <volmgt.h>
#include <sys/dkio.h>
#include <sys/fdio.h>
#include <sys/vtoc.h>
#include <sys/termios.h>
#include <sys/mount.h>
#include <ctype.h>
#include <signal.h>
#include <sys/wait.h>
#include <dirent.h>
#include <priv_utils.h>
#include <stdarg.h>
#include "rmformat.h"

/*
 * Definitions.
 */
#define SENSE_KEY(rqbuf)        (rqbuf[2] & 0xf) /* scsi error category */
#define ASC(rqbuf)              (rqbuf[12])     /* additional sense code */
#define ASCQ(rqbuf)             (rqbuf[13])     /* ASC qualifier */
```



```
#define DEFAULT_SCSI_TIMEOUT    60
#define INQUIRY_CMD             0x12
#define RQBUFLEN                32
#define CD_RW                   1               /* CD_RW/CD-R   */
#define WRITE_10_CMD            0x2A
#define READ_INFO_CMD           0x51
#define SYNC_CACHE_CMD          0x35
#define CLOSE_TRACK_CMD         0x5B
#define MODE_SENSE_10_CMD       0x5A
#define DEVFS_PREFIX            "/devices"

int             uscsi_error;            /* used for debugging failed uscsi */
char            rqbuf[RQBUFLEN];
static uint_t   total_retries;
static struct   uscsi_cmd uscmd;
static char     ucdb[16];
uchar_t         uscsi_status, rqstatus, rqresid;
int             total_devices_found = 0;
int             removable_found = 0;

extern char     *global_intr_msg;
extern int      vol_running;
extern char     *dev_name;
extern int32_t  m_flag;

/*
 * ON-private functions from libvolmgt
 */
int     _dev_mounted(char *path);

/*
 * Function prototypes.
 */
static int              my_umount(char *mountp);
static int              my_volrmmount(char *real_name);
static int              vol_name_to_dev_node(char *vname, char *found);
static int              vol_lookup(char *supplied, char *found);
static device_t         *get_device(char *user_supplied, char *node);
static char             *get_physical_name(char *path);
static int              lookup_device(char *supplied, char *found);
static void             fini_device(device_t *dev);
static int              is_cd(char *node);
void                    *my_zalloc(size_t size);
void                    err_msg(char *fmt, ...);
int                     inquiry(int fd, uchar_t *inq);
struct uscsi_cmd        *get_uscsi_cmd(void);
int                     uscsi(int fd, struct uscsi_cmd *scmd);
int                     get_mode_page(int fd, int page_no, int pc, int buf_len,
                            uchar_t *buffer);
int                     mode_sense(int fd, uchar_t pc, int dbd, int page_len,
                            uchar_t *buffer);
uint16_t                read_scsi16(void *addr);
int                     check_device(device_t *dev, int cond);
static void             get_media_info(device_t *t_dev, char *sdev,
                            char *pname, char *sn);

extern void             process_p_flag(smedia_handle_t handle, int32_t fd);

void
my_perror(char *err_string)
{

        int error_no;
        if (errno == 0)
                return;
```

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

*unchanged_portion_omitted_*

```
161 int32_t
162 check_and_unmount_vold(char *device_name, int32_t flag)
163 {
164         char *real_name;
165         char *nm;
166         char tmp_path_name[PATH_MAX];
167         struct stat stat_buf;
168         int32_t ret_val = 0;
169         struct  mnttab  *mntp;
170         FILE    *fp;
171         int nl;

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

175         if (volmgt_running() == 0) {
176                 DPRINTF("Vold not running\n");
177                 return (0);
178         }
179         if ((nm = volmgt_symname(device_name)) == NULL) {
180                 DPRINTF("path not managed\n");
181                 real_name = media_findname(device_name);
182         } else {
183                 DPRINTF1("path managed as %s\n", nm);
184                 real_name = media_findname(nm);
185                 DPRINTF1("real name %s\n", real_name);
186         }

188         if (real_name == NULL)
189                 return (-1);

191         /*
192          * To find out whether the device has been mounted by
193          * volume manager...
194          *
195          * Convert the real name to a block device address.
196          * Do a partial match with the mnttab entries.
197          * Make sure the match is in the beginning to avoid if
198          * anybody puts a label similiar to volume manager path names.
199          * Then use "volrmmount -e <dev_name>" if -U flag is set.
200          */

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

204         if (strncmp(real_name, "/vol/dev/", nl) != 0)
205                         return (0);
206         if (real_name[nl] == 'r') {
207                 (void) snprintf(tmp_path_name, PATH_MAX, "%s%s", "/vol/dev/",
208                     &real_name[nl + 1]);
209         } else {
210                 (void) snprintf(tmp_path_name, PATH_MAX, "%s", real_name);
211         }
212         DPRINTF1("%s \n", tmp_path_name);
213         ret_val = stat(tmp_path_name, &stat_buf);
214         if (ret_val < 0) {
215                 PERROR("Could not stat");
216                 return (-1);
217         }
```



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

221         if (fp == NULL) {
222                 PERROR("Could not open /etc/mnttab");
223                 return (-1);
224         }

226         mntp = (struct mnttab *)malloc(sizeof (struct mnttab));
227         if (mntp == NULL) {
228                 PERROR("malloc failed");
229                 (void) fclose(fp);
230                 return (-1);
231         }
232         errno = 0;
233         while (getmntent(fp, mntp) == 0) {
234                 if (errno != 0) {
235                         PERROR("Error with mnttab");
236                         (void) fclose(fp);
237                         return (-1);
238                 }
239                 /* Is it a probable entry? */
240                 DPRINTF1(" %s \n", mntp->mnt_special);
241                 if (strstr(mntp->mnt_special, tmp_path_name) !=
242                     mntp->mnt_special) {
243                         /* Skip to next entry */
244                         continue;
245                 } else {
246                         DPRINTF1("Found!! %s\n", mntp->mnt_special);
247                         ret_val = 1;
248                         break;
249                 }
250         }

252         if (ret_val == 1) {
253                 if (flag) {
254                         if (my_volrmmount(real_name) < 0) {
255                                 ret_val = -1;
256                         }
257                 } else {
258                         ret_val = -1;
259                 }
260         }
261         (void) fclose(fp);
262         free(mntp);
263         return (ret_val);
264 }
```

```
227 /*
228  * This routine checks if a device has mounted partitions. The
229  * device name is assumed to be /dev/rdsk/cNtNdNsN. So, this can
230  * be used for SCSI and PCMCIA cards.
231  * Returns
232  *       0 : if not mounted
233  *       1 : if successfully unmounted
234  *      -1 : Any error or umount failed
235  */

237 int32_t
238 check_and_unmount_scsi(char *device_name, int32_t flag)
239 {

241         struct  mnttab  *mntrefp;
242         struct  mnttab  *mntp;
243         FILE    *fp;
244         char block_dev_name[PATH_MAX];
```

```
245         char tmp_name[PATH_MAX];
246         int32_t  i, j;
247         int32_t unmounted = 0;

249         /*
250          * If the device name is not a character special, anyway we
251          * can not progress further
252          */

254         if (strncmp(device_name, "/dev/rdsk/c", strlen("/dev/rdsk/c")) != 0)
255                 return (0);

257         (void) snprintf(block_dev_name, PATH_MAX, "/dev/%s",
258             &device_name[strlen("/dev/r")]);
259         fp = fopen("/etc/mnttab", "r");

261         if (fp == NULL) {
262                 PERROR("Could not open /etc/mnttab");
263                 return (-1);
264         }

266         mntrefp = (struct mnttab *)malloc(sizeof (struct mnttab));
267         if (mntrefp == NULL) {
268                 PERROR("malloc failed");
269                 (void) fclose(fp);
270                 return (-1);
271         }

273         mntp = (struct mnttab *)malloc(sizeof (struct mnttab));
274         if (mntp == NULL) {
275                 PERROR("malloc failed");
276                 (void) fclose(fp);
277                 free(mntrefp);
278                 return (-1);
279         }

281         /* Try all the partitions */

283         (void) snprintf(tmp_name, PATH_MAX, "/dev/%s",
284             &device_name[strlen("/dev/r")]);

286         tmp_name[strlen("/dev/dsk/c0t0d0s")] = '\0';

288         errno = 0;
289         while (getmntent(fp, mntp) == 0) {
290                 if (errno != 0) {
291                         PERROR("Error with mnttab");
292                         (void) fclose(fp);
293                         return (-1);
294                 }
295                 /* Is it a probable entry? */
296                 if (strncmp(mntp->mnt_special, tmp_name, strlen(tmp_name))) {
297                         /* Skip to next entry */
298                         continue;
299                 }
300                 for (i = 0; i < NDKMAP; i++) {
301                         /* Check for ufs style mount devices */
302                         (void) snprintf(block_dev_name, PATH_MAX,
303                             "%s%d", tmp_name, i);

305                         if (strcmp(mntp->mnt_special, block_dev_name) == 0) {
306                                 if (flag) {
307                                         if (my_umount(mntp->mnt_mountp) < 0) {
308                                                 (void) fclose(fp);
309                                                 return (-1);
310                                         }
```



```
311                                         unmounted = 1;
312                                 } else {
313                                         (void) fclose(fp);
314                                         return (-1);
315                                 }
316                                 /* Skip to next entry */
317                                 continue;
318                         }

320                         /* Try for :1 -> :24 for pcfs */

322                         for (j = 1; j < 24; j++) {
323                                 (void) snprintf(block_dev_name, PATH_MAX,
324                                     "%s%d:%d", tmp_name, i, j);

326                                 if (strcmp(mntp->mnt_special,
327                                     block_dev_name) == 0) {
328                                         if (flag) {
329                                                 if (my_umount(mntp->mnt_mountp)
330                                                     < 0) {
331                                                         (void) fclose(fp);
332                                                         return (-1);
333                                                 }
334                                                 unmounted = 1;
335                                         } else {
336                                                 (void) fclose(fp);
337                                                 return (-1);
338                                         }
339                                         /* Skip to next entry */
340                                         continue;
341                                 }
342                                 (void) snprintf(block_dev_name, PATH_MAX,
343                                     "%s%d:%c", tmp_name, i, 'b' + j);

345                                 if (strcmp(mntp->mnt_special,
346                                     block_dev_name) == 0) {
347                                         if (flag) {
348                                                 if (my_umount(mntp->mnt_mountp)
349                                                     < 0) {
350                                                         (void) fclose(fp);
351                                                         return (-1);
352                                                 }
353                                                 unmounted = 1;
354                                         } else {
355                                                 (void) fclose(fp);
356                                                 return (-1);
357                                         }
358                                         /* Skip to next entry */
359                                         continue;
360                                 }
361                         }
362                 }

364         }

366         if (unmounted)
367                 return (1);
368         return (0);
369 }
```

_____***unchanged_portion_omitted_***

```
1005 static int
1006 my_volrmmount(char *real_name)
1007 {
1008         int pid, rval;
```

```
1010         /* Turn on the privileges. */
1011         (void) __priv_bracket(PRIV_ON);

1013         pid = fork();

1015         /* Turn off the privileges. */
1016         (void) __priv_bracket(PRIV_OFF);

1018         /* create a child to unmount the path */
1019         if (pid < 0) {
1020                 PERROR("fork failed");
1021                 exit(0);
1022         }

1024         if (pid == 0) {
1025                 /* the child */
1026                 /* get rid of those nasty err messages */
1027                 DPRINTF1("call_unmount_prog: calling %s \n",
1028                     "/usr/bin/volrmmount");

1030                 /* Turn on the privileges. */
1031                 (void) __priv_bracket(PRIV_ON);
1032                 if (execl("/usr/bin/volrmmount", "/usr/bin/volrmmount", "-e",
1033                     real_name, NULL) < 0) {
1034                         PERROR("volrmmount exec failed");
1035                         /* Turn off the privileges */
1036                         (void) __priv_bracket(PRIV_OFF);
1037                         exit(-1);
1038                 }
1039         } else if (waitpid(pid, &rval, 0) == pid) {
1040                 if (WIFEXITED(rval)) {
1041                         if (WEXITSTATUS(rval) == 0) {
1042                                 DPRINTF("volrmmount: Success\n");
1043                                 return (1);
1044                         }
1045                 }
1046         }
1047         return (-1);
1048 }
```

```
 966 int
 967 find_device(int defer, char *tmpstr)
 968 {
 969         DIR *dir;
 970         struct dirent *dirent;
 971         char sdev[PATH_MAX], dev[PATH_MAX], *pname;
 972         device_t *t_dev;
 973         int removable = 0;
 974         int device_type = 0;
 975         int hotpluggable = 0;
 976         struct dk_minfo mediainfo;
 977         static int found = 0;

 979         dir = opendir("/dev/rdsk");
 980         if (dir == NULL)
 981                 return (-1);

 983         total_devices_found = 0;
 984         while ((dirent = readdir(dir)) != NULL) {
 985                 if (dirent->d_name[0] == '.') {
 986                         continue;
 987                 }
 988                 (void) snprintf(sdev, PATH_MAX, "/dev/rdsk/%s",
 989                     dirent->d_name);
 990 #ifdef sparc
 991                 if (!strstr(sdev, "s2")) {
```



```
 992                         continue;
 993                 }
 994 #else /* x86 */
 995                 if (vol_running) {
 996                         if (!(strstr(sdev, "s2") || strstr(sdev, "p0"))) {
 997                                 continue;
 998                         }
 999                 } else {
1000                         if (!strstr(sdev, "p0")) {
1001                                 continue;
1002                         }
1003                 }
1004 #endif
1005                 if (!lookup_device(sdev, dev)) {
1006                         continue;
1007                 }
1008                 if ((t_dev = get_device(NULL, dev)) == NULL) {
1009                         continue;
1010                 }
1011                 total_devices_found++;

1013                 if ((!defer) && !found) {
1014                         char *sn, *tmpbuf = NULL;
1098                         char *sn, *tmpbuf;
1015                         /*
1016                          * dev_name is an optional command line input.
1017                          */
1018                         if (dev_name) {
1019                                 if (strstr(dirent->d_name, tmpstr)) {
1020                                         found = 1;
1021                                 } else if (!vol_running) {
1022                                         continue;
1023                                 }
1024                         }
1025                         /*
1026                          * volmgt_symname() returns NULL if the device
1027                          * is not managed by volmgt.
1028                          */
1029                         sn = volmgt_symname(sdev);

1031                         if (vol_running && (sn != NULL)) {
1032                                 if (strstr(sn, "dev") == NULL) {
1033                                         tmpbuf = (char *)my_zalloc(PATH_MAX);
1034                                         (void) strcpy(tmpbuf,
1035                                             "/vol/dev/aliases/");
1036                                         (void) strcat(tmpbuf, sn);
1037                                         free(sn);
1038                                         sn = tmpbuf;
1039                                 }
1040                                 if (dev_name && !found) {
1041                                         if (!strstr(tmpbuf, tmpstr)) {
1042                                                 continue;
1043                                         } else {
1044                                                 found = 1;
1045                                         }
1046                                 }
1047                         }

1049                         /*
1050                          * Get device type information for CD/DVD devices.
1051                          */
1052                         if (is_cd(dev)) {
1053                                 if (check_device(t_dev,
1054                                     CHECK_DEVICE_IS_DVD_WRITABLE)) {
1055                                         device_type = DK_DVDR;
1056                                 } else if (check_device(t_dev,
```

```
1057                                    CHECK_DEVICE_IS_DVD_READABLE)) {
1058                                        device_type = DK_DVDROM;
1059                                } else if (check_device(t_dev,
1060                                    CHECK_DEVICE_IS_CD_WRITABLE)) {
1061                                        device_type = DK_CDR;
1062                                } else {
1063                                        device_type = DK_CDROM;
1064                                }
1065                        } else {
1066                                device_type = ioctl(t_dev->d_fd,
1067                                    DKIOCGMEDIAINFO, &mediainfo);
1068                                if (device_type < 0)
1069                                        device_type = 0;
1070                                else
1071                                        device_type = mediainfo.dki_media_type;
1072                        }

1074                        if (!ioctl(t_dev->d_fd, DKIOCREMOVABLE, &removable) &&
1075                            !ioctl(t_dev->d_fd, DKIOCHOTPLUGGABLE,
1076                            &hotpluggable)) {
1077                                if (removable || hotpluggable) {
1078                                        removable_found++;
1079                                        pname = get_physical_name(sdev);
1080                                        if (sn) {
1081                                                (void) printf("  %4d. "
1082                                                    "Volmgt Node: %s\n",
1083                                                    removable_found, sn);
1084                                                (void) printf("        "
1085                                                    "Logical Node: %s\n", sdev);
1086                                                (void) printf("        "
1087                                                    "Physical Node: %s\n",
1088                                                    pname);
1089                                        } else {
1090                                                (void) printf("  %4d. "
1091                                                    "Logical Node: %s\n",
1092                                                    removable_found, sdev);
1093                                                (void) printf("        "
1094                                                    "Physical Node: %s\n",
1095                                                    pname);
1096                                        }
1097                                        (void) printf("        Connected "
1098                                            "Device: %-8.8s %-16.16s "
1099                                            "%-4.4s\n",
1100                                            &t_dev->d_inq[8],
1101                                            &t_dev->d_inq[16],
1102                                            &t_dev->d_inq[32]);
1103                                        (void) printf("        Device "
1104                                            "Type: ");
1105                                } else
1106                                        continue;
1107                        } else
1108                                continue;

1110                        switch (device_type) {
1111                                case DK_CDROM:
1112                                        (void) printf("CD Reader\n");
1113                                        break;
1114                                case DK_CDR:
1115                                case DK_CDRW:
1116                                        (void) printf("CD Reader/Writer\n");
1117                                        break;
1118                                case DK_DVDROM:
1119                                        (void) printf("DVD Reader\n");
1120                                        break;
1121                                case DK_DVDR:
1122                                case DK_DVDRAM:
```



```
1123                                        (void) printf("DVD Reader/Writer\n");
1124                                        break;
1125                                case DK_FIXED_DISK:
1126                                        if (strstr((const char *)
1127                                            &t_dev->d_inq[16], "FD") ||
1128                                            strstr((const char *)
1129                                            &t_dev->d_inq[16], "LS-120"))
1130                                                (void) printf("Floppy "
1131                                                    "drive\n");
1132                                        else
1133                                                (void) printf("Removable\n");
1134                                        break;
1135                                case DK_FLOPPY:
1136                                        (void) printf("Floppy drive\n");
1137                                        break;
1138                                case DK_ZIP:
1139                                        (void) printf("Zip drive\n");
1140                                        break;
1141                                case DK_JAZ:
1142                                        (void) printf("Jaz drive\n");
1143                                        break;
1144                                default:
1145                                        (void) printf("<Unknown>\n");
1146                                        DPRINTF1("\t   %d\n", device_type);
1147                                        break;
1148                        }
1149                        get_media_info(t_dev, sdev, pname, sn);
1150                }
1151                fini_device(t_dev);
1152        }

1154        (void) closedir(dir);
1155        return (removable_found);
1156 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   39066 Tue May 13 22:52:05 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>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 *
 *
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
 */

/*
 * rmf_slice.c :
 *      This file contains the functions for parsing a slice file
 *      for rmformat.
 */

#include <sys/types.h>
#include <ctype.h>
#include <sys/vtoc.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <memory.h>
#include <dirent.h>
#include <sys/fcntl.h>
#include <sys/param.h>
#include <sys/stat.h>
#include <stdio.h>
#include <sys/dkio.h>
#include <priv_utils.h>
#include "rmformat.h"

extern void my_perror(char *err_string);

static int32_t  last_token_type = 0;
#define spc()     (last_token_type)
```



```
/*
 * This global is used to store the current line # in the
 * data file. It must be global because the I/O routines
 * are allowed to side effect it to keep track of backslashed
 * newlines.
 */

static int32_t  data_lineno;            /* current line # in data file */

#define CHG_MODE_UNDEFINED  (-1)        /* undefined value */
#define CHG_MODE_SET    0               /* set bits by or'ing */
#define CHG_MODE_CLR    1               /* clr bits by and'ing */
#define CHG_MODE_ABS    2               /* set absolute value */


#define TOKEN_SIZE      36              /* max length of a token */
typedef char TOKEN[TOKEN_SIZE+1];       /* token type */
#define DATA_INPUT      0               /* 2 modes of input */
#define CMD_INPUT       1
#define WILD_STRING     "$"             /* wildcard character */
#define COMMENT_CHAR    '#'             /* comment character */

/*
 * List of strings with arbitrary matching values
 */
typedef struct slist {
        char    *str;
        char    *help;
        int32_t value;
} slist_t;

static slist_t ptag_choices[] = {
        { "unassigned", "",     V_UNASSIGNED    },
        { "boot",       "",     V_BOOT          },
        { "root",       "",     V_ROOT          },
        { "swap",       "",     V_SWAP          },
        { "usr",        "",     V_USR           },
        { "backup",     "",     V_BACKUP        },
        { "stand",      "",     V_STAND         },
        { "var",        "",     V_VAR           },
        { "home",       "",     V_HOME          },
        { "alternates", "",     V_ALTSCTR       },
        { NULL }
};


/*
 * Choices for the p_flag vtoc field
 */
static slist_t pflag_choices[] = {
        { "wm", "read-write, mountable",        0       },
        { "wu", "read-write, unmountable",      V_UNMNT },
        { "rm", "read-only, mountable",         V_RONLY },
        { "ru", "read-only, unmountable",       V_RONLY|V_UNMNT },
        { NULL }
};

/*
 * The definitions are the token types that the data file parser recognizes.
 */
#define SUP_EOF                 -1              /* eof token */
#define SUP_STRING              0               /* string token */
#define SUP_EQL                 1               /* equals token */
#define SUP_COMMA               2               /* comma token */
#define SUP_COLON               3               /* colon token */
```

```
#define SUP_EOL                 4               /* newline token */
#define SUP_OR                  5               /* vertical bar */
#define SUP_AND                 6               /* ampersand */
#define SUP_TILDE               7               /* tilde */


/*
 *      Prototypes for ANSI C compilers
 */
static int32_t  sup_prxfile(char *file_name, struct extvtoc *vt);
static int32_t  sup_setpart(struct extvtoc *vt);
static void     sup_pushchar(int32_t c);
static void     clean_token(char *cleantoken, char *token);
static void clean_token(char *cleantoken, char *token);
static int32_t sup_inputchar();
static int32_t sup_gettoken(char *buf);
static int32_t sup_get_token(char *buf);
static int32_t find_value(slist_t *slist, char *str, int32_t *value);
static int32_t check_vtoc_sanity(smedia_handle_t, int32_t fd,
                struct extvtoc *vt);
static uint64_t str2sector(char *str);
static int32_t strcnt(char *s1, char *s2);
static int32_t get_fdisk(smedia_handle_t, int32_t fd, int32_t offset,
                struct fdisk_info *fdisk);
static void erase(smedia_handle_t handle, diskaddr_t offset, diskaddr_t size);

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

static FILE *data_file;

static int32_t
sup_prxfile(char *file_name, struct extvtoc *vt)
{
        int32_t status, ret_val;
        TOKEN   token;
        TOKEN   cleaned;

        /*
         * Open the data file.  Return 0 if unable to do so.
         */
        data_file = fopen(file_name, "r");
        if (data_file == NULL) {
                PERROR("Open failed");
                return (-1);
        }
        /*
         * Step through the data file a meta-line at a time.  There are
         * typically several backslashed newlines in each meta-line,
         * so data_lineno will be getting side effected along the way.
         */
        data_lineno = 1;
        for (;;) {

                /*
                 * Get the keyword.
                 */
                status = sup_gettoken(token);
                /*
                 * If we hit the end of the data file, we're done.
                 */
                if (status == SUP_EOF)
                        break;
                /*
                 * If the line starts with some key character, it's an error.
```



```
                 */
                if (status != SUP_STRING) {
                        (void) fprintf(stderr,
                            gettext("Expecting keyword, found '%s'"),
                            token);
                        (void) fprintf(stderr,
                            gettext("Line no %d\n"), data_lineno);
                        continue;
                }
                /*
                 * Clean up the token and see which keyword it is.  Call
                 * the appropriate routine to process the rest of the line.
                 */
                clean_token(cleaned, token);
                if (strcmp(cleaned, "slices") == 0) {
                        ret_val = sup_setpart(vt);
                        (void) fclose(data_file);
                        return (ret_val);
                } else {
                        (void) fprintf(stderr, gettext("Unknown keyword '%s'"),
                            cleaned);
                        (void) fprintf(stderr,
                            gettext("Line no %d\n"), data_lineno);
                        (void) fclose(data_file);
                        return (-1);
                }
        }
        /*
         * Close the data file.
         */
        (void) fclose(data_file);

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

static int32_t
sup_gettoken(char *buf)
{
        /*
         * Skip end of lines and blank lines.
         */
        while ((last_token_type = sup_get_token(buf)) == SUP_EOL)
                ;
        return (last_token_type);
}

static int32_t
sup_get_token(char *buf)
{
        char    *ptr = buf;
        int32_t c, quoted = 0;

        /*
         * Was an end of file detected last try?
         */

        if (feof(data_file)) {
                return (SUP_EOF);
        }

        /*
         * Zero out the returned token buffer
         */
```

```
256         bzero(buf, TOKEN_SIZE + 1);

258         /*
259          * Strip off leading white-space.
260          */
261         while (isspace(c = sup_inputchar()))
262                 ;

264         /*
265          * Only white spaces and then end of file?
266          */

268         if (feof(data_file)) {
269                 return (SUP_EOF);
270         }

272         /*
273          * Read in characters until we hit unquoted white-space.
274          */
275         for (; !isspace(c) || quoted; c = sup_inputchar()) {

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                 }

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_EQL);
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;

369         /*
370          * Input the character.
371          */
372         c = getc(data_file);
373         /*
374          * If it's not a backslash, return it.
375          */

377         /*
378          * It was a backslash.  Get the next character.
379          */

381         if (c == '\\')
382                 c = getc(data_file);

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_PCMCIA_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_EQL) {
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                  */

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                  */

535                 /* Check for valid partition, e.g. > 8 or 16 */
536                 val1 = str2sector(cleaned);
537                 if (val1 == -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                 }

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);

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);

580                 /*
581                  * Pull in some grammar.
582                  */
583                 status = sup_gettoken(token);

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                         }


626                         if (status == SUP_COMMA) {
627                                 /* Can be tag only */

629                                 status = sup_gettoken(token);
630                                 if (status != SUP_STRING) {
631                                         (void) fprintf(stderr,
632                                             gettext("Expecting value"
633                                             ", found '%s'"),
634                                             token);
635                                         (void) fprintf(stderr,
636                                             gettext("Line no %d\n"),
637                                             data_lineno);
638                                         return (-1);
639                                 }

641                                 clean_token(cleaned, token);
642                                 if (find_value(ptag_choices,
643                                     cleaned, &i) == 1) {
644                                         DPRINTF1(" tag = %s", cleaned);
645                                         vtoc_tag = (ushort_t)i;
646                                 }
647                         status = sup_gettoken(token);
648                         }
649                 }

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                 }

676         }
677         return (0);
678 }

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;

688         nmatches = 0;
689         length = 0;

691         match_length = strlen(match_str);

693         for (; slist->str != NULL; slist++) {
694                 /*
695                  * See how many characters of the token match
696                  */
697                 i = strcnt(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 }

722 static int32_t
723 strcnt(char     *s1, char *s2)
724 {
725         int32_t i = 0;

727         while ((*s1 != '\0') && (*s1++ == *s2++))
728                 i++;
729         return (i);
730 }

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;

740         base = s2 = (char *)malloc(strlen(str) + 1);
741         if (s2 == NULL) {
742                 PERROR("Malloc failed");
743                 return (-1);
744         }
745         *s2 = '\0';



749         s1 = str;
750         while (*s1) {
751                 if ((*s1 != 'x') && ((*s1 < 'A') || (*s1 > 'F')) &&
752                     ((*s1 < 'a') || (*s1 > 'f')) && ((*s1 < '0') ||
753                     (*s1 > '9'))) {
754                         if (*s1 == 'G') {
755                                 mul_factor = 1024*1024*1024;
756                                 s1++;
757                         } else if (*s1 == 'M') {
758                                 mul_factor = 1024*1024;
759                                 s1++;
760                         } else if (*s1 == 'K') {
761                                 mul_factor = 1024;
762                                 s1++;
763                         }
764                         if ((*s1 != 'B') || (*(++s1) != NULL)) {
765                                 (void) fprintf(stderr,
766                                     gettext("Extra chars at the end\n"));
767                                 free(base);
768                                 return (-1);
769                         }
770                         break;
771                 } else {
772                         *s2++ = *s1++;
773                         *s2 = '\0';
774                 }
775         }
776         *s2 = NULL;

778         size = my_atoll(base);
779         if ((!mul_factor) || (size == -1)) {
780                 free(base);
781                 return (-1);
782         }
783         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;

808         vt->v_part[1].p_flag = V_UNMNT; /* Unmountable */
809         vt->v_part[2].p_flag = V_UNMNT; /* Unmountable */
810 #endif

812         ret_val = sup_prxfile(file_name, vt);
813         if (ret_val < 0)
814                 return (-1);

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 }

846 #define SWAP(a, b)      {diskaddr_t tmp; tmp = (a); (a) = (b); (b) = tmp; }

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  *                                   DKIOCGGEOM is sufficient.
860  * x86 : floppy, pcmcia : Don't check for fdisk. DKIOCGGEOM 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 {

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;
 24         uint32_t cyl_size;
890         int sparc_style = 0;    /* sparc_style handling ? */
891         struct fdisk_info fdisk;
892         int sol_part;
893         int total_parts = 0;

895 #ifdef sparc
896         sparc_style = 1;
897 #endif /* sparc */

899         if ((med_info.sm_media_type == SM_FLOPPY) ||
900             (med_info.sm_media_type == SM_PCMCIA_MEM) ||
901             (med_info.sm_media_type == SM_PCMCIA_ATA) ||
902             (med_info.sm_media_type == SM_SCSI_FLOPPY)) {
903                 sparc_style = 1;
904         }

906         if (sparc_style) {
907                 DPRINTF("sparc style true\n");
908                 if (ioctl(fd, DKIOCGGEOM, &dkg) < 0) {
909                         PERROR("DKIOCGGEOM Failed");
910                         return (-1);
911                 }
912                 media_size = (diskaddr_t)dkg.dkg_ncyl * dkg.dkg_nhead *
913                     dkg.dkg_nsect;
914                 cyl_size = dkg.dkg_nhead * dkg.dkg_nsect;
```

```
915         }

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 */

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;

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         }

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 */

973         if (media_size == 0) {
974                 media_size = (uint32_t)med_info.sm_capacity;
975         }

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                         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;

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                }

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 */

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_***

```
**********************************************************
   29524 Tue May 13 22:52:06 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>
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 *
 *
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2014 Andrew Stormont
#endif /* ! codereview */
 */

/*
 * Vold compatibility for rmvolmgr: emulate old commands as well as
 * action_filemgr.so to notify legacy apps via /tmp/.removable pipes.
 * A lot of this code is copied verbatim from vold sources.
 *
 * Here's the original description of action_filemgr.so:
 *
 * action_filemgr.so - filemgr interface routines for rmmount
 *
 * This shared object allows rmmount to communicate with filemgr.
 * This is done by communicating over a named pipe that filemgr
 * creates in directory NOTIFY_DIR.  The name of the pipe must
 * begin with NOTIFY_NAME.  This source file contains #define
 * compiler directives set the values of NOTIFY_DIR and NOTIFY_NAME.
 *
 * After a partition on a medium has been mounted as a result of
 * either insertion or remounting of the medium, the action()
 * method creates a file named with the symbolic name of the
 * device in which the medium is inserted and the partition name
 * (e.g. "jaz0-s2") in NOTIFY_DIR.  The file consists of one text
 * line containing a string naming the mount point of the partition,
 * a string giving the raw device path to the partition, and a
 * string naming the file system type on the partition.  The action()
 * method then sends a single character ('i' for insertion, 'r' for
 * remounting) through the named pipe NOTIFY_NAME to tell filemgr to
 * look for new files in NOTIFY_DIR.
 *
 * If a medium containing no mountable partitions is inserted
 * or remounted in a device, the action() method creates a file
```



```
 * named with the symbolic name of the device in NOTIFY_DIR.
 * The file consists of one text line containing a string
 * giving the symbolic name of the device and a string naming
 * the reason that the medium couldn't be mounted.  The action
 * method then sends either an 'i' or an 'r' through the named
 * pipe to tell filemgr to look for new files in NOTIFY_DIR.
 *
 * When a medium is ejected or unmounted, the action() method
 * removes the files that were created in NOTIFY_DIR when the medium
 * was inserted or remounted and sends a single character ('e' for
 * ejection, 'u' for unmounting) through the named pipe.
 *
 * The following environment variables must be set before calling action():
 *
 *      VOLUME_ACTION           action that occurred (e.g. "insert", "eject")
 *      VOLUME_SYMDEV           symbolic name (e.g. "cdrom0", "floppy1")
 *      VOLUME_NAME             volume name (e.g. "unnamed_cdrom", "s2")
 */


#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <strings.h>
#include <dirent.h>
#include <signal.h>
#include <errno.h>
#include <libintl.h>
#include <zone.h>
#include <pwd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/dkio.h>
#include <sys/cdio.h>
#include <sys/vtoc.h>
#include <sys/param.h>
#include <sys/wait.h>
#include <libcontract.h>
#include <sys/contract/process.h>
#include <sys/ctfs.h>
#include <tsol/label.h>

#include "vold.h"
#include "rmm_common.h"

int             rmm_debug = 0;
boolean_t       rmm_vold_actions_enabled = B_FALSE;
boolean_t       rmm_vold_mountpoints_enabled = B_FALSE;

static char     *prog_name = NULL;
static pid_t    prog_pid = 0;
static int      system_labeled = 0;
static uid_t    mnt_uid = (uid_t)-1;
static gid_t    mnt_gid = (gid_t)-1;
static zoneid_t mnt_zoneid = -1;
static char     mnt_zoneroot[MAXPATHLEN];
static char     mnt_userdir[MAXPATHLEN];

/*
 * Private attribute types and attributes.
 */
static const char notify_characters[] = {
        'e',
        '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     volrmmount_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_*

```
**********************************************************
    1974 Tue May 13 22:52:06 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>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 *
 *
 */
/*
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
 */

#ifndef _VOLD_H
#define _VOLD_H

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

#ifdef  __cplusplus
extern "C" {
#endif

#include <libhal.h>

typedef enum {
        EJECT,
        INSERT,
        REMOUNT,
        UNMOUNT,
        CLEAR_MOUNTS,
        CLOSETRAY
} action_t;
```

_____*unchanged_portion_omitted_*

```
extern int rmm_debug;
extern boolean_t rmm_vold_actions_enabled;
extern boolean_t rmm_vold_mountpoints_enabled;

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



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

#ifdef  __cplusplus
}
```

_____*unchanged_portion_omitted_*

```
**********************************************************
   13118 Tue May 13 22:52:06 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>
**********************************************************
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet
# at http://www.illumos.org/license/CDDL.
#

#
# Copyright 2011, Richard Lowe
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
# Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
#

include         $(SRC)/Makefile.master

MANSECT=        1

MANFILES=       acctcom.1                       \
                adb.1                           \
                addbib.1                        \
                alias.1                         \
                allocate.1                      \
                amt.1                           \
                appcert.1                       \
                apptrace.1                      \
                apropos.1                       \
                ar.1                            \
                arch.1                          \
                asa.1                           \
                at.1                            \
                atq.1                           \
                atrm.1                          \
                audioconvert.1                  \
                audioctl.1                      \
                audioplay.1                     \
                audiorecord.1                   \
                audiotest.1                     \
                auths.1                         \
                awk.1                           \
                banner.1                        \
                basename.1                      \
                bc.1                            \
                bdiff.1                         \
                bfs.1                           \
                break.1                         \
                builtin.1                       \
                cal.1                           \
                calendar.1                      \
                cancel.1                        \
                cat.1                           \
                cd.1                            \
                cdrw.1                          \
                checknr.1                       \
                chgrp.1                         \
```



```
                chkey.1                         \
                chmod.1                         \
                chown.1                         \
                ckdate.1                        \
                ckgid.1                         \
                ckint.1                         \
                ckitem.1                        \
                ckkeywd.1                       \
                ckpath.1                        \
                ckrange.1                       \
                ckstr.1                         \
                cksum.1                         \
                cktime.1                        \
                ckuid.1                         \
                ckyorn.1                        \
                clear.1                         \
                cmp.1                           \
                col.1                           \
                comm.1                          \
                command.1                       \
                compress.1                      \
                cp.1                            \
                cpio.1                          \
                cputrack.1                      \
                crle.1                          \
                crontab.1                       \
                crypt.1                         \
                csh.1                           \
                csplit.1                        \
                ctags.1                         \
                ctrun.1                         \
                ctstat.1                        \
                ctwatch.1                       \
                cut.1                           \
                date.1                          \
                dc.1                            \
                deallocate.1                    \
                deroff.1                        \
                dhcpinfo.1                      \
                diff.1                          \
                diff3.1                         \
                diffmk.1                        \
                digest.1                        \
                dircmp.1                        \
                dis.1                           \
                disown.1                        \
                dispgid.1                       \
                dispuid.1                       \
                dos2unix.1                      \
                download.1                      \
                dpost.1                         \
                du.1                            \
                dump.1                          \
                dumpcs.1                        \
                echo.1                          \
                ed.1                            \
                egrep.1                         \
                eject.1                         \
                elfdump.1                       \
                elfedit.1                       \
                elfsign.1                       \
                elfwrap.1                       \
                enable.1                        \
                encrypt.1                       \
                enhance.1                       \
                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                     getoptcvt.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                     indxbib.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                     madv.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                     mesg.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                     msgcvt.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                     ncab2clf.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                 rmmount.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                 volrmmount.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                 zonename.1

433 MANLINKS=       batch.1         \
434                 bg.1            \
435                 case.1          \
436                 chdir.1         \
437                 checkeq.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                 pdp11.1         \
495                 pfcsh.1         \
496                 pfiles.1        \
497                 pfksh.1         \
498                 pflags.1        \
499                 pfsh.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 errange.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
```

```
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
```



```
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.1has

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

748 .KEEP_STATE:

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

752 install:        $(ROOTMANFILES) $(ROOTMANLINKS)
```

```
**********************************************************
   11823 Tue May 13 22:52:06 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>
**********************************************************
```

```
'\" te
.\"  Copyright 2001, Sun Microsystems, Inc All Rights Reserved
.\"  Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH FDFORMAT 1 "May 2, 2014"
.TH FDFORMAT 1 "Feb 28, 2007"
.SH NAME
fdformat \- format floppy diskette or PCMCIA memory card
.SH SYNOPSIS
.LP
.nf
\fBfdformat\fR [\fB-dDeEfHlLmMUqvx\fR] [\fB-b\fR \fIlabel\fR] [\fB-B\fR \fIfilen
     [\fB-t\fR \fIdostype\fR] [\fIdevname\fR]
.fi

.SH DESCRIPTION
.sp
.LP
The \fBfdformat\fR utility has been superseded by \fBrmformat\fR(1), which
provides most but not all of \fBfdformat\fR's functionality.
.sp
.LP
\fBfdformat\fR is used to format diskettes and \fBPCMCIA\fR memory cards. All
new blank diskettes or \fBPCMCIA\fR memory cards must be formatted before they
can be used.
.sp
.LP
\fBfdformat\fR formats and verifies the media and indicates whether any bad
sectors were encountered. All existing data on the diskette or \fBPCMCIA\fR
memory card, if any, is destroyed by formatting. If no device name is given,
\fBfdformat\fR uses the diskette as a default.
.sp
.LP
By default, \fBfdformat\fR uses the configured capacity of the drive to format
the diskette. A \fB3.5\fR inch high-density drive uses diskettes with a
formatted capacity of \fB1.44MB\fR. A \fB5.25\fR inch high-density drive uses
diskettes with a formatted capacity of \fB1.2MB\fR. In either case, a density
option does not have to be specified to \fBfdformat\fR. However, a density
option must be specified when using a diskette with a lower capacity than the
drive's default. Use the \fB-H\fR option to format high-density diskettes
(\fB1.44MB\fR capacity) in an extra-high-density (\fBED\fR) drive. Use the
\fB-D\fR option, the \fB-l\fR option, or the \fB-L\fR option to format double-
density (or low-density) diskettes (\fB720KB\fR capacity) in an \fBHD\fR or
\fBED\fR drive. To format medium-density diskettes (\fB1.2MB\fR capacity), use
the \fB-M\fR option with \fB-t\fR \fBnec\fR (this is the same as using the
\fB-m\fR option with \fB-t\fR \fBnec\fR).
.sp
.LP
Extended density uses double-sided, extended-density or extra-high-density
(\fBDS\fR/\fBED\fR) diskettes. Medium and high densities use the same media:
double-sided, high-density (\fBDS\fR/\fBHD\fR) diskettes. Double (low) density
uses double-sided, double-density (\fBDS\fR/\fBDD\fR
.sp
.LP
\fBD\fR) diskettes. Substituting diskettes of one density for diskettes of
either a higher or lower density generally does not work. Data integrity cannot
```



```
be assured whenever a diskette is formatted to a capacity not matching its
density.
.sp
.LP
A \fBPCMCIA\fR memory card with densities from \fB512KB\fR to \fB64MB\fR may be
formatted.
.sp
.LP
\fBfdformat\fR writes new identification and data fields for each sector on all
tracks unless the \fB-x\fR option is specified. For diskettes, each sector is
verified if the \fB-v\fR option is specified.
.sp
.LP
After formatting and verifying, \fBfdformat\fR writes an operating-system label
on block \fB0\fR. Use the \fB-t\fR \fBdos\fR option (same as the \fB-d\fR
option) to put an \fBMS-DOS\fR file system on the diskette or \fBPCMCIA\fR
memory card after the format is done. Use the \fB-t\fR \fBnec\fR option with
the \fB-M\fR option (same as the \fB-m\fR option) to put an \fBNEC-DOS\fR file
system on a diskette. Otherwise, \fBfdformat\fR writes a \fBSunOS\fR label in
block \fB0\fR.
.SH OPTIONS
.sp
.LP
The following options are supported:
.sp
.ne 2
.na
\fB\fB-b\fR \fIlabel\fR\fR
.ad
.RS 15n
Labels the media with volume label. A SunOS volume label is restricted to 8
characters. A \fBDOS\fR volume label is restricted to 11 upper-case characters.
.RE

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

.sp
.ne 2
.na
\fB\fB-D\fR\fR
.ad
.RS 15n
Formats a \fB720KB\fR (3.5 inch) or \fB360KB\fR (5.25 inch) double-density
diskette (same as the \fB-l\fR or \fB-L\fR options). This is the default for
double-density type drives. It is needed if the drive is a high- or
extended-density type.
.RE

.sp
.ne 2
.na
\fB\fB-e\fR\fR
.ad
.RS 15n
Ejects the diskette when done. This feature is not available on all systems.
.RE
```

```
.sp
.ne 2
.na
\fB\fB-E\fR\fR
.ad
.RS 15n
Formats a \fB2.88MB\fR (3.5 inch) extended-density diskette. This is the
default for extended-density type drives.
.RE

.sp
.ne 2
.na
\fB\fB-f\fR\fR
.ad
.RS 15n
Forces formatting, that is, this option does not ask for confirmation before
starting format.
.RE

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

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

.sp
.ne 2
.na
\fB\fB-q\fR\fR
.ad
.RS 15n
Quiet; does not print status messages.
.RE

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

.sp
.ne 2
.na
\fB\fB-t\fR \fBnec\fR\fR
```



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

.sp
.ne 2
.na
\fB\fB-U\fR\fR
.ad
.RS 15n
Performs \fBumount\fR on any file systems and then formats. See
\fBmount\fR(1M).
.RE

.sp
.ne 2
.na
\fB\fB-v\fR\fR
.ad
.RS 15n
Verifies each block of the diskette after the format.
.RE

.sp
.ne 2
.na
\fB\fB-x\fR\fR
.ad
.RS 15n
Skips the format and only writes a SunOS label or an \fBMS-DOS\fR file system.
.RE

.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fIdevname\fR\fR
.ad
.RS 11n
Replaces \fIdevname\fR with \fBrdiskette0\fR (systems without volume
management) or \fBfloppy0\fR (systems with volume management) to use the first
drive or rdiskette1 (systems without volume management) or \fBfloppy1\fR
(systems with volume management) to use the second drive. If \fIdevname\fR is
omitted, the first drive, if one exists, is used. For \fBPCMCIA\fR memory
cards, replace \fIdevname\fR with the device name for the \fBPCMCIA\fR memory
card which resides in \fB/dev/rdsk/c\fIN\fRt\fIN\fRd\fIN\fRs\fIN\fR\fR or
/dev/dsk/c\fIN\fRt\fIN\fRd\fIN\fRs\fIN\fR. If \fIdevname\fR is omitted, the
default diskette drive, if one exists, is used.
.sp
If devname is omitted, the default diskette drive, if one exists, will be used.
\fIN\fR represents a decimal number and can be specified as follows:
.sp
.ne 2
.na
\fBc\fIN\fR\fR
.ad
.RS 6n
Controller \fIN\fR
.RE
```

```
.sp
.ne 2
.na
\fBt\fIN\fR\fR
.ad
.RS 6n
Technology type \fIN\fR:
.sp
.in +2
.nf

0x1       ROM
0x2       OTPROM
0x3       EPROM
0x4       EEPROM
0x5       FLASH
0x6       SRAM
0x7       DRAM
.fi
.in -2
.sp

.RE

.sp
.ne 2
.na
\fBd\fIN\fR\fR
.ad
.RS 6n
Technology region in type \fIN\fR.
.RE

.sp
.ne 2
.na
\fBs\fIN\fR\fR
.ad
.RS 6n
Slice \fIN\fR.
.RE

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

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

.sp
```



```
.ne 2
.na
\fB\fB-L\fR\fR
.ad
.RS 6n
Formats a \fB720KB\fR (3.5 inch) or \fB360KB\fR (5.25 inch) double-density
diskette (same as \fB-l\fR or \fB-D\fR). This is the default for double-density
type drives.
.RE

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

.RE

.SH FILES
.sp
.ne 2
.na
\fB/dev/diskette0\fR
.ad
.RS 24n
Directory providing block device access for the media in floppy drive \fB0\fR.
.RE

.sp
.ne 2
.na
\fB/dev/diskette0\fR
.ad
.RS 24n
Directory providing character device access for the media in floppy drive
\fB0\fR.
.RE

.sp
.ne 2
.na
\fB/dev/aliases/floppy0\fR
.ad
.RS 24n
Symbolic link to the character device for the media in floppy drive \fB0\fR.
.RE

.sp
.ne 2
.na
\fB/dev/rdiskette\fR
.ad
.RS 24n
Directory providing character device access for the media in the primary floppy
drive, usually drive \fB0\fR.
.RE

.sp
.ne 2
.na
\fB/dev/dsk/c\fIN\fRt\fIN\fRd\fIN\fRs\fIN\fR\fR
```

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

.sp
.ne 2
.na
\fB/dev/rdsk/c\fIN\fRt\fIN\fRd\fIN\fRs\fIN\fR\fR
.ad
.RS 24n
Directory providing character device access for the \fBPCMCIA\fR memory card.
See OPERANDS for a description of \fIN\fR.
.RE

.sp
.ne 2
.na
\fB/dev/aliases/pcmem\fIS\fR\fR
.ad
.RS 24n
Symbolic link to the character device for the \fBPCMCIA\fR memory card in
socket \fIS\fR where \fIS\fR represents a \fBPCMCIA\fR socket number.
.RE

.sp
.ne 2
.na
\fB/dev/rdsk/c\fIN\fRt\fIN\fRd\fIN\fRs\fIN\fR\fR
.ad
.RS 24n
Directory providing character device access for the \fBPCMCIA\fR memory card.
See OPERANDS for a description of \fIN\fR.
.RE

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

.SH SEE ALSO
.sp
.LP
\fBcpio\fR(1), \fBeject\fR(1), \fBrmformat\fR(1), \fBtar\fR(1),
\fBvolcheck\fR(1), \fBmount\fR(1M), \fBnewfs\fR(1M), \fBprtvtoc\fR(1M),
\fBattributes\fR(5), \fBpcfs\fR(7FS)
\fIvolcheck\fR(1), \fIvolrmmount\fR(1), \fImount\fR(1M), \fInewfs\fR(1M),
\fIprtvtoc\fR(1M), \fIattributes\fR(5), \fIpcfs\fR(7FS)
.SS "x86 Only"
.sp
.LP
\fBfd\fR(7D)
.SH NOTES
.sp
.LP
A diskette or \fBPCMCIA\fR memory card containing a \fBufs\fR file system
created on a SPARC based system (by using \fBfdformat\fR and \fBnewfs\fR(1M)),
is not identical to a diskette or \fBPCMCIA\fR memory card containing a ufs
file system created on an x86 based system. Do not interchange ufs diskettes or
memory cards between these platforms. Use \fBcpio\fR(1) or \fBtar\fR(1) to
```



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

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

```
**********************************************************
   14917 Tue May 13 22:52:07 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>
**********************************************************
'\" te
.\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved.
.\" Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH RMFORMAT 1 "May 2, 2014"
.TH RMFORMAT 1 "Feb 19, 2009"
.SH NAME
rmformat \- removable rewritable media format utility
.SH SYNOPSIS
.LP
.nf
\fBrmformat\fR [\fB-DeHUv\fR] [\fB-b\fR \fIlabel\fR] [\fB-c\fR \fIblockno\fR]
     [\fB-F\fRquick | long | force ] [\fB-s\fR \fIfilename\fR] [\fIdevname\fR]
.fi

.LP
.nf
\fBrmformat\fR \fB-V\fR read | write \fIdevname\fR
.fi

.LP
.nf
\fBrmformat\fR \fB-l\fR [\fIdevname\fR]
.fi

.SH DESCRIPTION
.sp
.LP
The \fBrmformat\fR utility is used to format, label, partition, and perform
other miscellaneous functions on removable, rewritable media that include
floppy drives, and the \fBPCMCIA\fR memory and \fBata\fR cards. The
\fBrmformat\fR utility should also be  used with all USB mass storage devices,
including USB hard drives. This utility can also be used for the verification
and surface analysis and for repair of the bad sectors found during
verification if the drive or the driver supports bad block management.
.sp
.LP
After formatting, \fBrmformat\fR writes the label, which covers the full
capacity of the media as one slice on floppy and \fBPCMCIA\fR memory cards to
maintain compatibility with the behavior of \fBfdformat\fR. The partition
information can be changed with the help of other options provided by
\fBrmformat\fR.
.SH OPTIONS
.sp
.LP
The following options are supported:
.sp
.ne 2
.na
\fB\fB-b\fR \fIlabel\fR\fR
.ad
.sp .6
.RS 4n
Labels the media with a SUNOS label. A SUNOS volume label name is restricted to
8 characters. For media size greater than 1 TB, an EFI label is created. For
writing a \fBDOS\fR Volume label, the user should use \fBmkfs_pcfs\fR(1M).
```



```
.RE

.sp
.ne 2
.na
\fB\fB-c\fR \fIblockno\fR\fR
.ad
.sp .6
.RS 4n
Corrects and repairs the given block. This correct and repair option may not be
applicable to all devices supported by \fBrmformat\fR, as some devices may have
a drive with bad block management capability and others may have this option
implemented in the driver. If the drive or driver supports bad block
management, a best effort is made to rectify the bad block. If the bad block
still cannot be rectified, a message is displayed to indicate the failure to
repair. The block number can be provided in decimal, octal, or hexadecimal
format.
.sp
The normal floppy and \fBPCMCIA\fR memory and ata cards do not support bad
block management.
.RE

.sp
.ne 2
.na
\fB\fB-D\fR\fR
.ad
.sp .6
.RS 4n
Formats a 720KB (3.5 inch) double density diskette. This is the default for
double density type drives. This option is needed if the drive is a high or
extended-density type.
.RE

.sp
.ne 2
.na
\fB\fB-e\fR\fR
.ad
.sp .6
.RS 4n
Ejects the media upon completion. This feature may not be available if the
drive does not support motorized eject.
.RE

.sp
.ne 2
.na
\fB\fB-F\fR quick | long | force\fR
.ad
.sp .6
.RS 4n
Formats the media.
.sp
The \fBquick\fR option starts a format without certification or format with
limited certification of certain tracks on the media.
.sp
The \fBlong\fR option starts a complete format. For some devices this might
include the certification of the whole media by the drive itself.
.sp
The \fBforce\fR option to format is provided to start a long format without
user confirmation before the format is  started.
.sp
In legacy media such as floppy drives, all options start a long format
depending on the mode (Extended Density mode, High Density mode, or Double
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 \fBru\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 \fBrmdisk\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/rdsk/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/rdsk/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/rdsk/c0t4d0s2:c\fR
343 # \fBmkfs -F pcfs /dev/rdsk/c0t4d0s2:c\fR
344 Construct a new FAT file system on /dev/rdsk/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/rdsk/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 \fBvolcheck\fR(1), \fBformat\fR(1M), \fBmkfs_pcfs\fR(1M), \fBmount\fR(1M),
 517 \fBnewfs\fR(1M), \fBprtvtoc\fR(1M), \fBrmmount\fR(1M), \fBrpc.smserverd\fR(1M),
 518 \fBattributes\fR(5), \fBscsa2usb\fR(7D), \fBsd\fR(7D), \fBpcfs\fR(7FS),
 519 \fBudfs\fR(7FS)
 511 \fBvolcheck\fR(1), \fBvolrmmount\fR(1), \fBformat\fR(1M), \fBmkfs_pcfs\fR(1M),
```

```
512 \fBmount\fR(1M), \fBnewfs\fR(1M), \fBprtvtoc\fR(1M), \fBrmmount\fR(1M),
513 \fBrpc.smserverd\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).
```

```
**********************************************************
    5890 Tue May 13 22:52:07 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>
**********************************************************
'\" te
.\"  Copyright (c) 2004, Sun Microsystems, Inc. All Rights Reserved
.\"  Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH RMMOUNT 1M "May 2, 2014"
.TH RMMOUNT 1M "Mar 1, 2007"
.SH NAME
rmmount \- removable media mounter for CD-ROM, floppy, Jaz drive, and others
.SH SYNOPSIS
.LP
.nf
\fB/usr/sbin/rmmount\fR [\fB-D\fR]
.fi

.SH DESCRIPTION
.sp
.LP
The \fBrmmount\fR utility is a removable media mounter that is executed by
volume management whenever a removable medium, such as a \fBCD-ROM\fR or a
floppy, is inserted. Removable media is managed by an application or a volume
manager. \fBrmmount\fR can also be called by using \fBvolrmmount\fR(1).
.sp
.LP
Upon insertion of a medium and following invocation of the \fBvolcheck\fR(1)
command, \fBrmmount\fR determines what type of file system (if any) is on that
medium. If a file system is present, \fBrmmount\fR mounts the file system in
one of the locations listed below.
.sp
.LP
For a diskette (floppy):
.sp
.ne 2
.na
\fB\fB/floppy/floppy0\fR\fR
.ad
.RS 26n
symbolic link to mounted floppy in local floppy drive
.RE

.sp
.ne 2
.na
\fB\fB/floppy/floppy_name\fR\fR
.ad
.RS 26n
mounted named floppy
.RE

.sp
.ne 2
.na
\fB\fB/floppy/unnamed_floppy\fR\fR
.ad
.RS 26n
mounted unnamed floppy
```



```
.RE

.sp
.LP
For a CD-ROM or a DVD-ROM:
.sp
.ne 2
.na
\fB\fB/cdrom/cdrom0\fR\fR
.ad
.sp .6
.RS 4n
symbolic link to mounted \fBCD-ROM\fR in local \fBCD-ROM\fR drive
.RE

.sp
.ne 2
.na
\fB\fB/cdrom/CD-ROM_name\fR\fR
.ad
.sp .6
.RS 4n
mounted named \fBCD-ROM\fR
.RE

.sp
.ne 2
.na
\fB\fB/cdrom/CD-ROM_name/partition\fR\fR
.ad
.sp .6
.RS 4n
mounted named \fBCD-ROM\fR with partitioned file system
.RE

.sp
.ne 2
.na
\fB\fB/cdrom/unnamed_cdrom\fR\fR
.ad
.sp .6
.RS 4n
mounted unnamed \fBCD-ROM\fR
.RE

.sp
.LP
For a Zip drive:
.sp
.ne 2
.na
\fB\fB/rmdisk/zip0\fR\fR
.ad
.RS 30n
symbolic link to mounted Zip medium in local Zip drive
.RE

.sp
.ne 2
.na
\fB\fB/rmdisk/\fIZip_name\fR\fR\fR
.ad
.RS 30n
mounted named Zip medium
.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, \fBrmmount\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, \fBrmmount\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 \fBvolcheck\fR(1), \fBfsck\fR(1M), \fBdlopen\fR(3C),
281 \fBvolcheck\fR(1), \fBvolrmmount\fR(1), \fBfsck\fR(1M), \fBdlopen\fR(3C),
287 \fBattributes\fR(5)
288 .sp
289 .LP
290 \fI\fR
```

```
**********************************************************
    2802 Tue May 13 22:52:07 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>
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2012 Nexenta Systems, Inc. All rights reserved.
# Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
#

set name=pkg.fmri value=pkg:/service/storage/media-volume-manager@$(PKGVERS)
set name=pkg.description value="Non-graphical removable volume manager"
set name=pkg.summary value="Removable volume manager"
set name=info.classification value=org.opensolaris.category.2008:System/Media
set name=variant.arch value=$(ARCH)
dir path=lib variant.opensolaris.zone=global
dir path=lib/svc variant.opensolaris.zone=global
dir path=lib/svc/manifest group=sys variant.opensolaris.zone=global
dir path=lib/svc/manifest/system group=sys variant.opensolaris.zone=global
dir path=lib/svc/manifest/system/filesystem group=sys \
    variant.opensolaris.zone=global
dir path=lib/svc/method variant.opensolaris.zone=global
dir path=usr group=sys
dir path=usr/bin
dir path=usr/lib
dir path=usr/sbin
dir path=usr/share/man/man1
dir path=usr/share/man/man1m
file path=lib/svc/manifest/system/filesystem/rmvolmgr.xml group=sys mode=0444 \
    variant.opensolaris.zone=global
file path=lib/svc/method/svc-rmvolmgr mode=0555 \
    variant.opensolaris.zone=global
file path=usr/bin/rmformat mode=4555
file path=usr/bin/rmmount mode=0555
file path=usr/bin/volcheck mode=0555
file path=usr/bin/volrmmount mode=0555
file path=usr/lib/rmvolmgr mode=0555
file path=usr/share/man/man1/rmformat.1
file path=usr/share/man/man1/rmmount.1
file path=usr/share/man/man1/volcheck.1
file path=usr/share/man/man1/volrmmount.1
```



```
file path=usr/share/man/man1m/rmmount.1m
file path=usr/share/man/man1m/rmvolmgr.1m
legacy pkg=SUNWrmvolmgr desc="Non-graphical removable volume manager" \
    name="Removable volume manager"
legacy pkg=SUNWrmvolmgrr desc="Non-graphical removable volume manager (Root)" \
    name="Removable volume manager (Root)"
license cr_Sun license=cr_Sun
license lic_CDDL license=lic_CDDL
link path=usr/bin/rmumount target=./rmmount
link path=usr/sbin/rmmount target=../bin/rmmount
link path=usr/share/man/man1/rmumount.1 target=rmmount.1
```