****
1061 Sat Feb 8 11:08:22 2020
new/usr/src/cmd/getfacl/Makefile
12288 getfacl and setfacl could stand improvement
****

```
#
# 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 (c) 1993,2001 by Sun Microsystems, Inc.
# All rights reserved.
#
# Copyright 2020 Peter Tribble.
#

PROG= getfacl

include ../Makefile.cmd

CERRWARN += -_gcc=-Wno-unused-variable
CERRWARN += $(CNOWARN_UNINIT)

.KEEP_STATE:

all: $(PROG)

install: all $(ROOTPROG)

clean:

lint:   lint_PROG

include ../Makefile.targ
```

```
**********************************************************
    7056 Sat Feb  8 11:08:22 2020
new/usr/src/cmd/getfacl/getfacl.c
12288 getfacl and setfacl could stand improvement
**********************************************************
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2020 Peter Tribble.
 */

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

#ifndef lint
static char sccsid[] = "%Z%%M%  %I%     %E% SMI";
#endif

/*
 * getfacl [-ad] file ...
 * This command displays discretionary information for a file or files.
 * display format:
 *      # file: filename
 *      # owner: uid
 *      # group: gid
 *      user::perm
 *      user:uid:perm
 *      group::perm
 *      group:gid:perm
 *      mask:perm
 *      other:perm
 *      default:user::perm
 *      default:user:uid:perm
 *      default:group::perm
 *      default:group:gid:perm
 *      default:mask:perm
 *      default:other:perm
 */

#include <stdlib.h>
#include <stdio.h>
#include <pwd.h>
#include <grp.h>
#include <locale.h>
#include <sys/acl.h>
#include <errno.h>
```

```
static char     *pruname(uid_t);
static char     *prgname(gid_t);
static char     *display(int);
static void     usage();


int
main(int argc, char *argv[])
{
        int             c;
        int             aflag = 0;
        int             dflag = 0;
        int             errflag = 0;
        int             savecnt;
        int             aclcnt;
        int             mask = 0;
        int             mask;
        aclent_t        *aclp;
        aclent_t        *tp;
        char            *permp;

        (void) setlocale(LC_ALL, "");
        (void) textdomain(TEXT_DOMAIN);

        if (argc < 2)
                usage();

        while ((c = getopt(argc, argv, "ad")) != EOF) {
                switch (c) {
                case 'a':
                        aflag++;
                        break;
                case 'd':
                        dflag++;
                        break;
                case '?':
                        errflag++;
                        break;
                }
        }
        if (errflag)
                usage();

        if (optind >= argc)
                usage();

        for (; optind < argc; optind++) {
                register char *filep;

                filep = argv[optind];

                /* Get ACL info of the files */
                errno = 0;
                if ((aclcnt = acl(filep, GETACLCNT, 0, NULL)) < 0) {
                        if (errno == ENOSYS) {
                                (void) fprintf(stderr,
                                    gettext("File system doesn't support "
                                    "aclent_t style ACL's.\n"
                                    "See acl(5) for more information on "
                                    "POSIX-draft ACL support.\n"));
                                    "Solaris ACL support.\n"));
                                exit(2);
                        }
                        perror(filep);
                        exit(2);
                }
```

```
		if (aclcnt < MIN_ACL_ENTRIES) {
			(void) fprintf(stderr,
			    gettext("%d: acl count too small from %s\n"),
			    aclcnt, filep);
			exit(2);
		}

		if ((aclp = (aclent_t *)malloc(sizeof (aclent_t) * aclcnt))
		    == NULL) {
			(void) fprintf(stderr,
			    gettext("Insufficient memory\n"));
			exit(1);
		}

		errno = 0;
		if (acl(filep, GETACL, aclcnt, aclp) < 0) {
			perror(filep);
			exit(2);
		}

		/* display ACL: assume it is sorted. */
		(void) printf("\n# file: %s\n", filep);
		savecnt = aclcnt;
		for (tp = aclp; aclcnt--; tp++) {
			if (tp->a_type == USER_OBJ)
				(void) printf("# owner: %s\n",
				    pruname(tp->a_id));
			if (tp->a_type == GROUP_OBJ)
				(void) printf("# group: %s\n",
				    prgname(tp->a_id));
			if (tp->a_type == CLASS_OBJ)
				mask = tp->a_perm;
		}
		aclcnt = savecnt;
		for (tp = aclp; aclcnt--; tp++) {
			switch (tp->a_type) {
			case USER:
				if (!dflag) {
					permp = display(tp->a_perm);
					(void) printf("user:%s:%s\t\t",
					    pruname(tp->a_id), permp);
					free(permp);
					permp = display(tp->a_perm & mask);
					(void) printf(
					    "#effective:%s\n", permp);
					free(permp);
				}
				break;
			case USER_OBJ:
				if (!dflag) {
					/* no need to display uid */
					permp = display(tp->a_perm);
					(void) printf("user::%s\n", permp);
					free(permp);
				}
				break;
			case GROUP:
				if (!dflag) {
					permp = display(tp->a_perm);
					(void) printf("group:%s:%s\t\t",
					    prgname(tp->a_id), permp);
					free(permp);
					permp = display(tp->a_perm & mask);
					(void) printf(
					    "#effective:%s\n", permp);
					free(permp);
```

```
				}
				break;
			case GROUP_OBJ:
				if (!dflag) {
					permp = display(tp->a_perm);
					(void) printf("group::%s\t\t", permp);
					free(permp);
					permp = display(tp->a_perm & mask);
					(void) printf(
					    "#effective:%s\n", permp);
					free(permp);
				}
				break;
			case CLASS_OBJ:
				if (!dflag) {
					permp = display(tp->a_perm);
					(void) printf("mask:%s\n", permp);
					free(permp);
				}
				break;
			case OTHER_OBJ:
				if (!dflag) {
					permp = display(tp->a_perm);
					(void) printf("other:%s\n", permp);
					free(permp);
				}
				break;
			case DEF_USER:
				if (!aflag) {
					permp = display(tp->a_perm);
					(void) printf("default:user:%s:%s\n",
					    pruname(tp->a_id), permp);
					free(permp);
				}
				break;
			case DEF_USER_OBJ:
				if (!aflag) {
					permp = display(tp->a_perm);
					(void) printf("default:user::%s\n",
					    permp);
					free(permp);
				}
				break;
			case DEF_GROUP:
				if (!aflag) {
					permp = display(tp->a_perm);
					(void) printf("default:group:%s:%s\n",
					    prgname(tp->a_id), permp);
					free(permp);
				}
				break;
			case DEF_GROUP_OBJ:
				if (!aflag) {
					permp = display(tp->a_perm);
					(void) printf("default:group::%s\n",
					    permp);
					free(permp);
				}
				break;
			case DEF_CLASS_OBJ:
				if (!aflag) {
					permp = display(tp->a_perm);
					(void) printf("default:mask:%s\n",
					    permp);
					free(permp);
				}
				break;
			case DEF_OTHER_OBJ:
				if (!aflag) {
					permp = display(tp->a_perm);
					(void) printf("default:other:%s\n",
					    permp);
					free(permp);
				}
				break;
			default:
				(void) fprintf(stderr,
				    gettext("unrecognized entry\n"));
				break;
			}
		}
		free(aclp);
	}
	return (0);
}
```
______***unchanged_portion_omitted_***

_**_
1084 Sat Feb 8 11:08:23 2020
new/usr/src/cmd/setfacl/Makefile
12288 getfacl and setfacl could stand improvement
_**_

```
#
# 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 (c) 1993 by Sun Microsystems, Inc.
#
# Copyright (c) 2018, Joyent, Inc.
# Copyright 2020 Peter Tribble.
#

PROG= setfacl

include ../Makefile.cmd

CERRWARN += -_gcc=-Wno-unused-variable
CERRWARN += -_gcc=-Wno-implicit-function-declaration
CERRWARN += $(CNOWARN_UNINIT)

# not linted
SMATCH=off

LDLIBS += -lsec

.KEEP_STATE:

all: $(PROG)

install: all $(ROOTPROG)

clean:

lint:   lint_PROG

include ../Makefile.targ
```

```
**********************************************************
   20408 Sat Feb  8 11:08:23 2020
new/usr/src/cmd/setfacl/setfacl.c
12288 getfacl and setfacl could stand improvement
**********************************************************
/*
 * 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) 1993, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2020 Peter Tribble.
 */

/*
 * setfacl [-r] -f aclfile file ...
 * setfacl [-r] -d acl_entries file ...
 * setfacl [-r] -m acl_entries file ...
 * setfacl [-r] -s acl_entries file ...
 * This command deletes/adds/modifies/sets discretionary information for a file
 * or files.
 */

#include <stdlib.h>
#include <stdio.h>
#include <pwd.h>
#include <grp.h>
#include <string.h>
#include <locale.h>
#include <sys/acl.h>
#include <sys/types.h>
#include <unistd.h>
#include <errno.h>
#include <ctype.h>

#define ADD     1
#define MODIFY  2
#define DELETE  3
#define SET     4

static int get_acl_info(char *filep, aclent_t **aclpp);
static int mod_entries(aclent_t *, int, char *, char *, char *, int);
static int set_file_entries(char *, char *, int);
static int set_online_entries(char *, char *, int);
static void usage();
static int parse_entry_list(aclent_t **, int *, char *, int);
static int convert_to_aclent_t(char *, int *, aclent_t **, int);
static int parse_entry(char *, aclent_t *, int);
static void err_handle(int, aclent_t *);
static int conv_id(char *);
```

```
int
main(int argc, char *argv[])
{
        int             c;
        int             dflag = 0;
        int             mflag = 0;
        int             rflag = 0;
        int             sflag = 0;
        int             fflag = 0;
        int             errflag = 0;
        int             aclcnt;                 /* used by -m -d */
        aclent_t        *aclp;                  /* used by -m -d */
        char            *aclfilep = NULL;               /* acl file argument */
        char            *aclfilep;              /* acl file argument */
        char            *d_entryp = NULL;       /* ptr to del entry list */
        char            *m_entryp = NULL;       /* ptr to mod entry list */
        char            *s_entryp = NULL;       /* ptr to set entry list */
        char            *work_dp = NULL;        /* working ptrs for the above */
        char            *work_mp = NULL;
        char            *work_sp = NULL;

        (void) setlocale(LC_ALL, "");
        (void) textdomain(TEXT_DOMAIN);

        if (argc < 3)
                usage();

        while ((c = getopt(argc, argv, "rm:d:s:f:")) != EOF) {
                switch (c) {
                case 'r':
                        rflag++;
                        break;
                case 'd':
                        if (dflag || fflag || sflag)
                                usage();
                        dflag++;
                        d_entryp = optarg;
                        break;
                case 'm':
                        if (mflag || fflag || sflag)
                                usage();
                        mflag++;
                        m_entryp = optarg;
                        break;
                case 's':
                        if (fflag || sflag || mflag || dflag)
                                usage();
                        sflag++;
                        s_entryp = optarg;
                        break;
                case 'f':
                        if (fflag || sflag || mflag || dflag)
                                usage();
                        fflag++;
                        aclfilep = optarg;
                        break;
                case '?':
                        errflag++;
                        break;
                }
        }
        if (errflag)
                usage();

        /* one of these flags should be set */
        if (!fflag && !sflag && !mflag && !dflag)
                usage();

        /* no file arguments */
        if (optind >= argc)
                usage();

        for (; optind < argc; optind++) {
                register char *filep;

                filep = argv[optind];

                /* modify and delete: we need to get the ACL first */
                if (mflag || dflag) {
                        if (m_entryp != NULL) {
                                free(work_mp);
                                work_mp = strdup(m_entryp);
                                if (work_mp == NULL) {
                                        fprintf(stderr,
                                            gettext("out of memory %s\n"),
                                            m_entryp);
                                        exit(1);
                                }
                        }

                        if (d_entryp != NULL) {
                                free(work_dp);
                                work_dp = strdup(d_entryp);
                                if (work_dp == NULL) {
                                        fprintf(stderr,
                                            gettext("out of memory %s\n"),
                                            d_entryp);
                                        exit(1);
                                }
                        }

                        aclcnt = get_acl_info(filep, &aclp);
                        if (aclcnt == -1)
                                exit(2);
                        if (mod_entries(aclp, aclcnt, work_mp,
                            work_dp, filep, rflag) == -1)
                                exit(2);
                } else if (fflag) {
                        if (set_file_entries(aclfilep, filep, rflag) == -1)
                                exit(2);
                } else if (sflag) {
                        if (s_entryp != NULL) {
                                free(work_sp);
                                work_sp = strdup(s_entryp);
                                if (work_sp == NULL) {
                                        fprintf(stderr,
                                            gettext("out of memory %s\n"),
                                            s_entryp);
                                        exit(1);
                                }
                        }
                        if (set_online_entries(work_sp, filep, rflag) == -1)
                                exit(2);
                }
        }
        return (0);
}

/*
 * For add, modify, and delete, we need to get the ACL of the file first.
 */
```

```
static int
get_acl_info(char *filep, aclent_t **aclpp)
{
        int     aclcnt;

        if ((aclcnt = acl(filep, GETACLCNT, 0, NULL)) < 0) {
                if (errno == ENOSYS) {
                        (void) fprintf(stderr,
                            gettext("File system doesn't support aclent_t "
                            "style ACL's.\n"
                            "See acl(5) for more information on"
                            "POSIX-draft ACL support.\n"));
                            " ACL styles support by Solaris.\n"));
                        return (-1);
                }
                (void) fprintf(stderr,
                    gettext("%s: failed to get acl count\n"), filep);
                perror("get acl count error");
                return (-1);
        }
        if (aclcnt < MIN_ACL_ENTRIES) {
                (void) fprintf(stderr,
                    gettext("%d: acl count is too small from %s\n"),
                    aclcnt, filep);
                return (-1);
        }

        if ((*aclpp = (aclent_t *)malloc(sizeof (aclent_t) * aclcnt)) == NULL) {
                (void) fprintf(stderr, gettext("out of memory\n"));
                return (-1);
        }
        if (acl(filep, GETACL, aclcnt, *aclpp) < 0) {
                (void) fprintf(stderr,
                    gettext("%s: failed to get acl entries\n"), filep);
                perror("getacl error");
                return (-1);
        }
        return (aclcnt);
}

/*
 * mod_entries() handles add, delete, and modify ACL entries of a file.
 * The real action is in convert_to_aclent_t() called by parse_entry_list().
 * aclp: points ACL of a file and may be changed by lower level routine.
 * modp: modify entry list in ascii format
 * delp: delete entry list in ascii format
 * fnamep: file of interest
 */
static int
mod_entries(aclent_t *aclp, int cnt, char *modp, char *delp,
    char *fnamep, int rfg)
{
        int     rc;             /* return code */

        /* modify and add: from -m option */
        if (parse_entry_list(&aclp, &cnt, modp, MODIFY) == -1)
                return (-1);

        /* deletion: from -d option */
        if (parse_entry_list(&aclp, &cnt, delp, DELETE) == -1)
                return (-1);

        if (aclsort(cnt, rfg, aclp) == -1) {
                (void) err_handle(cnt, aclp);
                (void) fprintf(stderr,
                    gettext("aclcnt %d, file %s\n"), cnt, fnamep);
                return (-1);
        }

        if (acl(fnamep, SETACL, cnt, aclp) < 0) {
                fprintf(stderr,
                    gettext("%s: failed to set acl entries\n"), fnamep);
                perror("setacl error");
                return (-1);
        }
        return (0);
}
______unchanged_portion_omitted_

/*
 * set_online_entries() parses the acl entries from command line (setp).
 * It converts the comma separated acl entries into aclent_t format.
 * It then recalculates the mask according to rflag.
 * Finally it sets ACL to the file (fnamep).
 */
static int
set_online_entries(char *setp, char *fnamep, int rflag)
{
        char            *commap;
        aclent_t        *aclp;
        int             aclcnt = 0;

        if (parse_entry_list(&aclp, &aclcnt, setp, SET) == -1)
                return (-1);

        if (aclsort(aclcnt, rflag, aclp) == -1) {
                (void) err_handle(aclcnt, aclp);
                (void) fprintf(stderr,
                    gettext("aclcnt %d, file %s\n"), aclcnt, fnamep);
                return (-1);
        }

        if (acl(fnamep, SETACL, aclcnt, aclp) < 0) {
                fprintf(stderr,
                    gettext("%s: failed to set acl entries\n"), fnamep);
                perror("setacl error");
                return (-1);
        }
        return (0);
}
______unchanged_portion_omitted_

/*
 * convert_to_aclent_t() converts an acl entry in ascii format (fields separated
 * by colon) into aclent_t and appends it to the current ACL. It also handles
 * memory allocation/deallocation for acl entries in aclent_t format.
 * aclpp that contains acl entries in acl format will be returned.
 * We don't check duplicates.
 */
static int
convert_to_aclent_t(char *entryp, int *cntp, aclent_t **aclpp, int mode)
{
        aclent_t        *new_aclp;
        aclent_t        tmpacl;
        aclent_t        *taclp, *centry = NULL, *gentry = NULL;
        int             cur_cnt;
        int             found = 0;
        int             is_obj;

        if (entryp == NULL)
                return (0);
```

```
        if (*cntp > 1)
                new_aclp = (aclent_t *)realloc(*aclpp,
                    sizeof (aclent_t) * (*cntp));
        else
                new_aclp = (aclent_t *) malloc(sizeof (aclent_t) * (*cntp));
        if (new_aclp == NULL) {
                fprintf(stderr,
                    gettext("Insufficient memory for acl %d\n"), *cntp);
                return (-1);
        }

        tmpacl.a_id = 0;        /* id field needs to be initialized */
        if (entryp[0] == 'u')
                tmpacl.a_id = getuid(); /* id field for user */
        if (entryp[0] == 'g')
                tmpacl.a_id = getgid(); /* id field for group */

        tmpacl.a_type = 0;
        if (parse_entry(entryp, &tmpacl, mode) == -1)
                return (-1);

        is_obj = ((tmpacl.a_type == USER_OBJ) ||
            (tmpacl.a_type == GROUP_OBJ) ||
            (tmpacl.a_type == CLASS_OBJ) ||
            (tmpacl.a_type == DEF_USER_OBJ) ||
            (tmpacl.a_type == DEF_GROUP_OBJ) ||
            (tmpacl.a_type == DEF_OTHER_OBJ));

        if (*cntp > 1)
                new_aclp = (aclent_t *)realloc(*aclpp,
                    sizeof (aclent_t) * (*cntp));
        else
                new_aclp = (aclent_t *) malloc(sizeof (aclent_t) * (*cntp));
        if (new_aclp == NULL) {
                fprintf(stderr,
                    gettext("Insufficient memory for acl %d\n"), *cntp);
                return (-1);
        }

        cur_cnt = *cntp - 1;
        switch (mode) {
        case MODIFY:    /* and add */
                for (taclp = new_aclp; cur_cnt-- > 0; taclp++) {
                        if (taclp->a_type == tmpacl.a_type &&
                            ((taclp->a_id == tmpacl.a_id) || is_obj)) {
                                found++;
                                /* cnt is added before it's called */
                                *cntp -= 1;
                                taclp->a_perm = tmpacl.a_perm;
                                break;
                        }
                }
                if (!found)     /* Add it to the end: no need to change cntp */
                        memcpy(new_aclp + *cntp -1, &tmpacl, sizeof (aclent_t));
                break;

        case DELETE:
                for (taclp = new_aclp; cur_cnt-- > 0; taclp++) {
                        if (taclp->a_type == tmpacl.a_type &&
                            ((taclp->a_id == tmpacl.a_id) || is_obj)) {
                                found++;
                                /* move up the rest */
                                while (cur_cnt-- > 0) {
                                        memcpy(taclp, taclp+1,
                                            sizeof (aclent_t));
                                        taclp++;
                                }
                                *cntp = *cntp - 2;
                                break;
                        }
                }
                if (!found)
                        *cntp -= 1;
                break;

        case SET:
                /* we may check duplicate before copying over?? */
                memcpy(new_aclp + *cntp -1, &tmpacl, sizeof (aclent_t));
                break;

        default:
                fprintf(stderr,
                    gettext("Unrecognized mode: internal error\n"));
                break;
        }

        /*
         * If converting from non-trivial acl entry to trivial one,
         * reset CLASS_OBJ's permission with that of GROUP_OBJ.
         */

        if (mode == DELETE) {
                boolean_t       trivial = B_TRUE;       /* assumption */
                cur_cnt = *cntp;
                for (taclp = new_aclp; cur_cnt-- > 0; taclp++) {
                        switch (taclp->a_type) {
                                case USER_OBJ:
                                case OTHER_OBJ:
                                        break;
                                case CLASS_OBJ:
                                        centry = taclp;
                                        break;
                                case GROUP_OBJ:
                                        gentry = taclp;
                                        break;
                                default:
                                        /*
                                         * Confirmed that the new acl set is
                                         * still a non-trivial acl.
                                         * Skip reset.
                                         */
                                        trivial = B_FALSE;
                        }
                }
                if (centry != NULL && gentry != NULL && trivial == B_TRUE)
                        centry->a_perm = gentry->a_perm;
        }
        *aclpp = new_aclp;      /* return new acl entries */
        return (0);
}
```
______***unchanged_portion_omitted_***

```
************************************************************
    7842 Sat Feb  8 11:08:23 2020
new/usr/src/man/man1/getfacl.1
12288 getfacl and setfacl could stand improvement
************************************************************
'\" te
.\" \&.Copyright (c) 2002, Sun Microsystems, Inc.  All Rights Reserved
.\" Copyright (c) 2020 Peter Tribble.
.\" 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 GETFACL 1 "Feb 8, 2020"
.TH GETFACL 1 "Nov 5, 1994"
.SH NAME
getfacl \- display discretionary file information
.SH SYNOPSIS
.LP
.nf
\fBgetfacl\fR [\fB-ad\fR] \fIfile\fR...
.fi

.SH DESCRIPTION
.sp
.LP
For each argument that is a regular file, special file, or named pipe, the
\fBgetfacl\fR utility displays the owner, the group, and the Access Control
List (\fBACL\fR). For each directory argument, \fBgetfacl\fR displays the
owner, the group, and the \fBACL\fR and/or the default \fBACL\fR. Only
directories contain default \fBACL\fRs.
.sp
.LP
The \fBgetfacl\fR utility will fail if executed on a file system that supports
NFSv4 \fBACL\fRs.  See \fBacl\fR(5) for a description of the difference
between the older POSIX-draft \fBACL\fRs and the newer NFSv4 \fBACL\fRs.  The
\fBls\fR(1) utility, when used with the \fB-v\fR or \fB-V\fR options, will
display \fBACL\fRs on all types of file system.
.sp
.LP
The \fBgetfacl\fR utility may be executed on a file system that does not
support \fBACL\fRs. It reports the \fBACL\fR based on the base permission bits.
.sp
.LP
With no options specified, \fBgetfacl\fR displays the filename, the file owner,
the file group owner, and both the \fBACL\fR and the default \fBACL\fR, if it
exists.
.SH OPTIONS
.sp
.LP
The following options are supported:
.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.RS 6n
Displays the filename, the file owner, the file group owner, and the \fBACL\fR
of the file.
.RE

.sp
.ne 2
.na
\fB\fB-d\fR\fR
.ad
.RS 6n
Displays the filename, the file owner, the file group owner, and the default
```

```
\fBACL\fR of the file, if it exists.
.RE

.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fIfile\fR\fR
.ad
.RS 8n
The path name of a regular file, special file, or named pipe.
.RE

.SH OUTPUT
.sp
.LP
The format for \fBACL\fR output is as follows:
.sp
.in +2
.nf
# file: filename
# owner: uid
# group: gid
user::perm
user:uid:perm
group::perm
group:gid:perm
mask:perm
other:perm
default:user::perm
default:user:uid:perm
default:group::perm
default:group:gid:perm
default:mask:perm
default:other:perm
.fi
.in -2
.sp

.sp
.LP
When multiple files are specified on the command line, a blank line separates
the \fBACL\fRs for each file.
.sp
.LP
The \fBACL\fR entries are displayed in the order in which they are evaluated
when an access check is performed. The default \fBACL\fR entries that may exist
on a directory have no effect on access checks.
.sp
.LP
The first three lines display the filename, the file owner, and the file group
owner. Notice that when only the \fB-d\fR option is specified and the file has
no default \fBACL\fR, only these three lines are displayed.
.sp
.LP
The \fBuser\fR entry without a user \fBID\fR indicates the permissions that
are granted to the file owner. One or more additional user entries indicate the
permissions that are granted to the specified users.
.sp
.LP
The \fBgroup\fR entry without a group \fBID\fR indicates the permissions that
are granted to the file group owner. One or more additional group entries
indicate the permissions that are granted to the specified groups.
```

```
.sp
.LP
The \fBmask\fR entry indicates the \fBACL\fR mask permissions. These are the
maximum permissions allowed to any user entries except the file owner, and to
any group entries, including the file group owner. These permissions restrict
the permissions specified in other entries.
.sp
.LP
The \fBother\fR entry indicates the permissions that are granted to others.
.sp
.LP
The \fBdefault\fR entries may exist only for directories. These entries
indicate the default entries that are added to a file created within the
directory.
.sp
.LP
The \fBuid\fR is a login name or a user \fBID\fR if there is no entry for the
\fBuid\fR in the system password file, \fB/etc/passwd\fR. The \fBgid\fR is a
group name or a group \fBID\fR if there is no entry for the \fBgid\fR in the
system group file, \fB/etc/group\fR. The \fBperm\fR is a three character string
composed of the letters representing the separate discretionary access rights:
\fBr\fR (read), \fBw\fR (write), \fBx\fR (execute/search), or the place holder
character \fB\(mi\fR\&. The \fBperm\fR is displayed in the following order:
\fBrwx\fR. If a permission is not granted by an \fBACL\fR entry, the place
holder character appears.
.sp
.LP
If you use the \fBchmod\fR(1) command to change the file group owner
permissions on a file with \fBACL\fR entries, both the file group owner
permissions and the \fBACL\fR mask are changed to the new permissions. Be aware
that the new \fBACL\fR mask permissions may change the effective permissions
for additional users and groups who have \fBACL\fR entries on the file.
.sp
.LP
In order to indicate that the \fBACL\fR mask restricts an \fBACL\fR entry,
\fBgetfacl\fR displays an additional tab character, pound sign (\fB#\fR), and
the actual permissions granted, following the entry.
.SH EXAMPLES
.LP
\fBExample 1 \fRDisplaying file information
.sp
.LP
Given file \fBfoo\fR, with an \fBACL\fR six entries long, the command

.sp
.in +2
.nf
host% \fBgetfacl foo\fR
.fi
.in -2
.sp

.sp
.LP
would print:

.sp
.in +2
.nf
# file: foo
# owner: shea
# group: staff
user::rwx
user:spy:\|\(mi\|\(mi\|\(mi
user:mookie:r\|\(mi\|\(mi
group::r\|\(mi\|\(mi
```

```
mask::rw\|\(mi
other::\|\(mi\|\(mi\|\(mi
.fi
.in -2
.sp

.LP
\fBExample 2 \fRDisplaying information after chmod command
.sp
.LP
Continue with the above example, after \fBchmod\fR \fB700 foo\fR was issued:

.sp
.in +2
.nf
host% \fBgetfacl foo\fR
.fi
.in -2
.sp

.sp
.LP
would print:

.sp
.in +2
.nf
# file: foo
# owner: shea
# group: staff
user::rwx
user:spy:\|\(mi\|\(mi\|\(mi
user:mookie:r\|\(mi\|\(mi     #effective:\|\(mi\|\(mi\|\(mi
group::\|\(mi\|\(mi\|\(mi
mask::\|\(mi\|\(mi\|\(mi
other::\|\(mi\|\(mi\|\(mi
.fi
.in -2
.sp

.LP
\fBExample 3 \fRDisplaying information when ACL contains default entries
.sp
.LP
Given directory \fBdoo\fR, with an \fBACL\fR containing default entries, the
command

.sp
.in +2
.nf
host% \fBgetfacl -d doo\fR
.fi
.in -2
.sp

.sp
.LP
would print:

.sp
.in +2
.nf
# file: doo
# owner: shea
# group: staff
default:user::rwx
```

```
default:user:spy:\|\(mi\|\(mi\|\(mi
default:user:mookie:r\|\(mi\|\(mi
default:group::r\|\(mi\|\(mi
default:mask::\|\(mi\|\(mi\|\(mi
default:other::\|\(mi\|\(mi\|\(mi
.fi
.in -2
.sp

.SH FILES
.sp
.ne 2
.na
\fB\fB/etc/passwd\fR\fR
.ad
.RS 15n
system password file
.RE

.sp
.ne 2
.na
\fB\fB/etc/group\fR\fR
.ad
.RS 15n
group file
.RE

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Evolving
.TE

.SH SEE ALSO
.sp
.LP
\fBchmod\fR(1), \fBls\fR(1), \fBsetfacl\fR(1), \fBacl\fR(2),
\fBaclsort\fR(3SEC), \fBgroup\fR(4), \fBpasswd\fR(4), \fBacl\fR(5),
\fBattributes\fR(5)
\fBaclsort\fR(3SEC), \fBgroup\fR(4), \fBpasswd\fR(4), \fBattributes\fR(5)
.SH NOTES
.sp
.LP
The output from \fBgetfacl\fR is in the correct format for input to the
\fBsetfacl\fR \fB-f\fR command. If the output from \fBgetfacl\fR is redirected
to a file, the file may be used as input to \fBsetfacl\fR. In this way, a user
may easily assign one file’s \fBACL\fR to another file.
```

```
**********************************************************
   11823 Sat Feb  8 11:08:23 2020
new/usr/src/man/man1/setfacl.1
12288 getfacl and setfacl could stand improvement
**********************************************************
’\" te
.\"  Copyright (c) 2006, Sun Microsystems, Inc. All Rights Reserved
.\" Copyright (c) 2020 Peter Tribble.
.\" 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 SETFACL 1 "Feb 8, 2020"
.TH SETFACL 1 "Dec 19, 2006"
.SH NAME
setfacl \- modify the Access Control List (ACL) for a file or files
.SH SYNOPSIS
.LP
.nf
\fBsetfacl\fR [\fB-r\fR] \fB-s\fR \fIacl_entries\fR \fIfile\fR
.fi

.LP
.nf
\fBsetfacl\fR [\fB-r\fR] \fB-md\fR \fIacl_entries\fR \fIfile\fR
.fi

.LP
.nf
\fBsetfacl\fR [\fB-r\fR] \fB-f\fR \fIacl_file\fR \fIfile\fR
.fi

.SH DESCRIPTION
.sp
.LP
For each file specified, \fBsetfacl\fR either replaces its entire \fBACL\fR,
including the default \fBACL\fR on a directory, or it adds, modifies, or
deletes one or more \fBACL\fR entries, including default entries on
directories.
.sp
.LP
The \fBsetfacl\fR utility can only manipulate POSIX-draft \fBACL\fRs.  See
\fBacl\fR(5) for a description of the difference between the older POSIX-draft
\fBACL\fRs and the newer NFSv4 \fBACL\fRs.  The \fBchmod\fR(1) utility can
be used to manipulate \fBACL\fRs on all types of file system.
.sp
.LP
When the \fBsetfacl\fR command is used, it can result in changes to the file
permission bits. When the user \fBACL\fR entry for the file owner is changed,
the file owner class permission bits are modified. When the group \fBACL\fR
entry for the file group class is changed, the file group class permission bits
are modified. When the other \fBACL\fR entry is changed, the file other class
permission bits are modified.
.sp
.LP
If you use the \fBchmod\fR(1) command to change the file group owner
permissions on a file with \fBACL\fR entries, both the file group owner
permissions and the \fBACL\fR mask are changed to the new permissions. Be aware
that the new \fBACL\fR mask permissions can change the effective permissions
for additional users and groups who have \fBACL\fR entries on the file.
.sp
.LP
A directory can contain default \fBACL\fR entries. If a file or directory is
created in a directory that contains default \fBACL\fR entries, the newly
created file has permissions generated according to the intersection of the
default \fBACL\fR entries and the permissions requested at creation time. The
\fBumask\fR(1) are not applied if the directory contains default \fBACL\fR
```

```
entries. If a default \fBACL\fR is specified for a specific user (or users),
the file has a regular \fBACL\fR created. Otherwise, only the mode bits are
initialized according to the intersection described above. The default
\fBACL\fR should be thought of as the maximum discretionary access permissions
that can be granted.
.sp
.LP
Use the \fBsetfacl\fR command to set ACLs on files in a UFS file system, which
supports POSIX-draft ACLS (or \fBaclent_t\fR style ACLs). Use the \fBchmod\fR
command to set ACLs on files in a ZFS file system, which supports NFSv4-style
ACLS (or \fBace_t\fR style ACLs).
.SS "\fIacl_entries\fR Syntax"
.sp
.LP
For the \fB-m\fR and \fB-s\fR options, \fIacl_entries\fR are one or more
comma-separated \fBACL\fR entries.
.sp
.LP
An \fBACL\fR entry consists of the following fields separated by colons:
.sp
.ne 2
.na
\fB\fIentry_type\fR\fR
.ad
.RS 14n
Type of \fBACL\fR entry on which to set file permissions. For example,
\fIentry_type\fR can be \fBuser\fR (the owner of a file) or \fBmask\fR (the
\fBACL\fR mask).
.RE

.sp
.ne 2
.na
\fB\fIuid\fR or \fIgid\fR\fR
.ad
.RS 14n
User name or user identification number. Or, group name or group identification
number.
.RE

.sp
.ne 2
.na
\fB\fIperms\fR\fR
.ad
.RS 14n
Represents the permissions that are set on \fIentry_type\fR. \fIperms\fR can be
indicated by the symbolic characters \fBrwx\fR or a number (the same
permissions numbers used with the \fBchmod\fR command).
.RE

.sp
.LP
The following table shows the valid \fBACL\fR entries (default entries can only
be specified for directories):
.sp

.sp
.TS
c c
l l .
\fBACL\fR Entry Description
_
u[ser]::\fIperms\fR     File owner permissions.
g[roup]::\fIperms\fR    File group owner permissions.
o[ther]:\fIperms\fR     T{
```

```
Permissions for users other than the file owner or members of file group owner.
T}
m[ask]:\fIperms\fR      T{
The \fBACL\fR mask. The mask entry indicates the maximum permissions allowed for
T}
u[ser]:\fIuid:perms\fR  T{
Permissions for a specific user. For \fIuid\fR, you can specify either a user na
T}
g[roup]:\fIgid:perms\fR T{
Permissions for a specific group. For \fIgid\fR, you can specify either a group
T}
d[efault]:u[ser]::\fIperms\fR   Default file owner permissions.
d[efault]:g[roup]::\fIperms\fR  Default file group owner permissions.
d[efault]:o[ther]:\fIperms\fR   T{
Default permissions for users other than the file owner or members of the file g
T}
d[efault]:m[ask]:\fIperms\fR    Default \fBACL\fR mask.
d[efault]:u[ser]:\fIuid\fR:\fIperms\fR  T{
Default permissions for a specific user. For \fIuid\fR, you can specify either a
T}
d[efault]:g[roup]:\fIgid\fR:\fIperms\fR T{
Default permissions for a specific group. For \fIgid\fR, you can specify either
T}
.TE

.sp
.LP
For the \fB-d\fR option, \fIacl_entries\fR are one or more comma-separated
\fBACL\fR entries without permissions. Notice that the entries for file owner,
file group owner, \fBACL\fR mask, and others can not be deleted.
.SH OPTIONS
.sp
.LP
The options have the following meaning:
.sp
.ne 2
.na
\fB\fB-d\fR \fIacl_entries\fR\fR
.ad
.RS 18n
Deletes one or more entries from the file. The entries for the file owner, the
file group owner, and others can not be deleted from the \fBACL\fR. Notice that
deleting an entry does not necessarily have the same effect as removing all
permissions from the entry.
.RE

.sp
.ne 2
.na
\fB\fB-f\fR \fIacl_file\fR\fR
.ad
.RS 18n
Sets a file's \fBACL\fR with the \fBACL\fR entries contained in the file named
\fIacl_file\fR. The same constraints on specified entries hold as with the
\fB-s\fR option. The entries are not required to be in any specific order in
the file. Also, if you specify a dash (\fB-\fR) for \fIacl_file\fR, standard
input is used to set the file's \fBACL\fR.
.sp
The character \fB#\fR in \fIacl_file\fR can be used to indicate a comment. All
characters, starting with the \fB#\fR until the end of the line, are ignored.
Notice that if the \fIacl_file\fR has been created as the output of the
\fBgetfacl\fR(1) command, any effective permissions, which follow a \fB#\fR,
are ignored.
.RE

.sp
```

```
.ne 2
.na
\fB\fB-m\fR \fIacl_entries\fR\fR
.ad
.RS 18n
Adds one or more new \fBACL\fR entries to the file, and/or modifies one or more
existing \fBACL\fR entries on the file. If an entry already exists for a
specified \fIuid\fR or \fIgid\fR, the specified permissions replace the current
permissions. If an entry does not exist for the specified \fIuid\fR or
\fIgid\fR, an entry is created. When using the \fB-m\fR option to modify a
default \fBACL\fR, you must specify a complete default \fBACL\fR (user, group,
other, mask, and any additional entries) the first time.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.RS 18n
Recalculates the permissions for the \fBACL\fR mask entry. The permissions
specified in the \fBACL\fR mask entry are ignored and replaced by the maximum
permissions necessary to grant the access to all additional user, file group
owner, and additional group entries in the \fBACL\fR. The permissions in the
additional user, file group owner, and additional group entries are left
unchanged.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR \fIacl_entries\fR\fR
.ad
.RS 18n
Sets a file's \fBACL\fR. All old \fBACL\fR entries are removed and replaced
with the newly specified \fBACL\fR. The entries need not be in any specific
order. They are sorted by the command before being applied to the file.
.sp
Required entries:
.RS +4
.TP
.ie t \(bu
.el o
Exactly one \fBuser\fR entry specified for the file owner.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Exactly one \fBgroup\fR entry for the file group owner.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Exactly one \fBother\fR entry specified.
.RE
If there are additional user and group entries:
.RS +4
.TP
.ie t \(bu
.el o
Exactly one \fBmask\fR entry specified for the \fBACL\fR mask that indicates
the maximum permissions allowed for users (other than the owner) and groups.
.RE
.RS +4
```

```
.TP
.ie t \(bu
.el o
Must not be duplicate \fBuser\fR entries with the same \fIuid\fR.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Must not be duplicate \fBgroup\fR entries with the same \fIgid\fR.
.RE
If \fIfile\fR is a directory, the following default \fBACL\fR entries can be
specified:
.RS +4
.TP
.ie t \(bu
.el o
Exactly one \fBdefault user\fR entry for the file owner.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Exactly one \fBdefault group\fR entry for the file group owner.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Exactly one \fBdefault mask\fR entry for the \fBACL\fR mask.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Exactly one \fBdefault other\fR entry.
.RE
There can be additional \fBdefault user\fR entries and additional \fBdefault
group\fR entries specified, but there can not be duplicate additional
\fBdefault user\fR entries with the same \fIuid\fR, or duplicate \fBdefault
group\fR entries with the same \fIgid\fR.
.RE

.SH EXAMPLES
.LP
\fBExample 1 \fRAdding read permission only
.sp
.LP
The following example adds one \fBACL\fR entry to file \fBabc\fR, which gives
user \fBshea\fR read permission only.

.sp
.in +2
.nf
\fBsetfacl -m user:shea:r\(mi\(mi abc\fR
.fi
.in -2
.sp

.LP
\fBExample 2 \fRReplacing a file’s entire \fBACL\fR
.sp
.LP
The following example replaces the entire \fBACL\fR for the file \fBabc\fR,
which gives \fBshea\fR read access, the file owner all access, the file group
owner read access only, the \fBACL\fR mask read access only, and others no
```

```
access.

.sp
.in +2
.nf
\fBsetfacl -s user:shea:rwx,user::rwx,group::rw-,mask:r--,other:--- abc\fR
.fi
.in -2
.sp

.sp
.LP
Notice that after this command, the file permission bits are \fBrwxr-----\fR.
Even though the file group owner was set with read/write permissions, the
\fBACL\fR mask entry limits it to have only read permission. The mask entry
also specifies the maximum permissions available to all additional user and
group \fBACL\fR entries. Once again, even though the user \fBshea\fR was set
with all access, the mask limits it to have only read permission. The \fBACL\fR
mask entry is a quick way to limit or open access to all the user and group
entries in an \fBACL\fR. For example, by changing the mask entry to read/write,
both the file group owner and user \fBshea\fR would be given read/write access.

.LP
\fBExample 3 \fRSetting the same \fBACL\fR on two files
.sp
.LP
The following example sets the same \fBACL\fR on file \fBabc\fR as the file
\fBxyz\fR.

.sp
.in +2
.nf
\fBgetfacl xyz | setfacl -f \(mi abc\fR
.fi
.in -2
.sp

.SH FILES
.sp
.ne 2
.na
\fB\fB/etc/passwd\fR\fR
.ad
.RS 15n
password file
.RE

.sp
.ne 2
.na
\fB\fB/etc/group\fR\fR
.ad
.RS 15n
group file
.RE

.SH SEE ALSO
.sp
.LP
\fBchmod\fR(1), \fBgetfacl\fR(1), \fBumask\fR(1), \fBaclcheck\fR(3SEC),
\fBaclsort\fR(3SEC), \fBgroup\fR(4), \fBpasswd\fR(4), \fBacl\fR(5),
\fBattributes\fR(5)
\fBaclsort\fR(3SEC), \fBgroup\fR(4), \fBpasswd\fR(4), \fBattributes\fR(5)
```

```
************************************************************
   11921 Sat Feb  8 11:08:23 2020
new/usr/src/man/man3sec/acl_totext.3sec
12288 getfacl and setfacl could stand improvement
************************************************************
'\" te
.\" Copyright (c) 20068 Sun Microsystems, Inc. All Rights Reserved.
.\" 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 ACL_TOTEXT 3SEC "Jun 16, 2008"
.SH NAME
acl_totext, acl_fromtext \- convert internal representation  to or from
external representation
.SH SYNOPSIS
.LP
.nf
cc [ \fIflag\fR\&.\|.\|. ] \fIfile\fR\&.\|.\|. \fB-lsec\fR [ \fIlibrary\fR\&.\|.
#include <sys/acl.h>

\fBchar *\fR\fBacl_totext\fR(\fBacl_t *\fR\fIaclp\fR, \fBint\fR \fIflags\fR);
.fi

.LP
.nf
\fBint\fR \fBacl_fromtext\fR(\fBchar *\fR\fIacltextp\fR, \fBacl_t **\fR\fIaclp\f
.fi

.SH DESCRIPTION
.sp
.LP
The \fBacl_totext()\fR function converts an internal ACL representation pointed
to by \fIaclp\fR into an external ACL representation. The memory for the
external text string is obtained using \fBmalloc\fR(3C). The caller is
responsible for freeing the memory upon completion.
.sp
.LP
The format of the external ACL is controlled by the \fIflags\fR argument.
Values for \fIflags\fR are constructed by a bitwise-inclusive-OR of \fIflags\fR
from the following list, defined in <\fBsys/acl.h\fR>.
.sp
.ne 2
.na
\fB\fBACL_COMPACT_FMT\fR\fR
.ad
.RS 19n
For NFSv4 ACLs, the ACL entries will be formatted using the compact ACL format
detailed in \fBls\fR(1) for the \fB-V\fR option.
.RE

.sp
.ne 2
.na
\fB\fBACL_APPEND_ID\fR\fR
.ad
.RS 19n
Append the \fBuid\fR or \fBgid\fR for additional user or group entries.  This
flag is used to construct ACL entries in a manner that is suitable for archive
flag is used to construt ACL entries in a manner that is suitable for archive
utilities such as \fBtar\fR(1). When the ACL is translated from the external
format to internal representation using \fBacl_fromtext()\fR, the appended ID
will be used to populate the \fBuid\fR or \fBgid\fR field of the ACL entry when
the user or group name does not exist on the host system. The appended id will
be ignored when the user or group name does exist on the system.
.RE
```

```
.sp
.ne 2
.na
\fB\fBACL_SID_FMT\fR\fR
.ad
.RS 19n
For NFSv4 ACLs, the ACL entries for user or group entries will use the
\fBusersid\fR or \fBgroupsid\fR format when the "id" field in the ACL entry is
an ephemeral \fBuid\fR or \fBgid\fR.  The raw \fBsid\fR format will only be
used when the "id" cannot be resolved to a windows name.
.RE

.sp
.LP
The \fBacl_fromtext()\fR function converts an external ACL representation
pointed to by \fIacltextp\fR into an internal ACL representation. The memory
for the list of ACL entries is obtained using \fBmalloc\fR(3C). The caller is
responsible for freeing the memory upon completion. Depending on type of ACLs a
file system supports, one of two external external representations are
possible. For POSIX draft file systems such as ufs, the external representation
is described in \fBacltotext\fR(3SEC). The external ACL representation For
NFSv4-style ACLs is detailed as follows.
.sp
.LP
Each \fBacl_entry\fR contains one ACL entry. The external representation of an
ACL entry contains three, four or five colon separated fields. The first field
contains the ACL entry type. The entry type keywords are defined as:
.sp
.ne 2
.na
\fB\fBeveryone@\fR\fR
.ad
.RS 13n
This ACL entry specifies the access granted to any user or group that does not
match any previous ACL entry.
.RE

.sp
.ne 2
.na
\fB\fBgroup\fR\fR
.ad
.RS 13n
This ACL entry with a GID specifies the access granted to a additional group of
the object.
.RE

.sp
.ne 2
.na
\fB\fBgroup@\fR\fR
.ad
.RS 13n
This ACL entry with no GID specified in the ACL entry field specifies the
access granted to the owning group of the object.
.RE

.sp
.ne 2
.na
\fB\fBgroupsid\fR\fR
.ad
.RS 13n
This ACL entry with a SID or Windows name specifies the access granted to a
Windows group. This type of entry is for a CIFS server created file.
.RE
```

```
.sp
.ne 2
.na
\fB\fBowner@\fR\fR
.ad
.RS 13n
This ACL entry with no UID specified in the ACL entry field specifies the
access granted to the owner of the object.
.RE

.sp
.ne 2
.na
\fB\fBsid\fR\fR
.ad
.RS 13n
This ACL entry with a SID or Windows name when the entry could be either a
group or a user.
.RE

.sp
.ne 2
.na
\fB\fBuser\fR\fR
.ad
.RS 13n
This ACL entry with a UID specifies the access granted to a additional user of
the object.
.RE

.sp
.ne 2
.na
\fB\fBusersid\fR\fR
.ad
.RS 13n
This ACL entry with a SID or Windows name specifies the access granted to a
Windows user. This type of entry is for a CIFS server created file.
.RE

.sp
.LP
The second field contains the ACL entry ID, and is used only for user or group
ACL entries. This field is not used for \fBowner@\fR, \fBgroup@\fR, or
\fBeveryone@\fR entries.
.sp
.ne 2
.na
\fB\fBuid\fR\fR
.ad
.RS 7n
This field contains a user-name or user-ID. If the user-name cannot be resolved
to a UID, then the entry is assumed to be a numeric UID.
.RE

.sp
.ne 2
.na
\fB\fBgid\fR\fR
.ad
.RS 7n
This field contains a group-name or group-ID. If the group-name can’t be
resolved to a GID, then the entry is assumed to be a numeric GID.
.RE
```

```
.sp
.LP
The third field contains the discretionary access permissions. The format of
the permissions depends on whether \fBACL_COMPACT_FMT\fR is specified. When the
\fIflags\fR field does not request \fBACL_COMPACT_FMT\fR, the following format
is used with a forward slash (/) separating the permissions.
.sp
.ne 2
.na
\fB\fBadd_file\fR\fR
.ad
.RS 20n
Add a file to a directory.
.RE

.sp
.ne 2
.na
\fB\fBadd_subdirectory\fR\fR
.ad
.RS 20n
Add a subdirectory.
.RE

.sp
.ne 2
.na
\fB\fBappend\fR\fR
.ad
.RS 20n
Append data.
.RE

.sp
.ne 2
.na
\fB\fBdelete\fR\fR
.ad
.RS 20n
Delete.
.RE

.sp
.ne 2
.na
\fB\fBdelete_child\fR\fR
.ad
.RS 20n
Delete child.
.RE

.sp
.ne 2
.na
\fB\fBexecute\fR\fR
.ad
.RS 20n
Execute permission.
.RE

.sp
.ne 2
.na
\fB\fBlist_directory\fR\fR
.ad
.RS 20n
```

```
List a directory.
.RE

.sp
.ne 2
.na
\fB\fBread_acl\fR\fR
.ad
.RS 20n
Read ACL.
.RE

.sp
.ne 2
.na
\fB\fBread_data\fR\fR
.ad
.RS 20n
Read permission.
.RE

.sp
.ne 2
.na
\fB\fBread_attributes\fR\fR
.ad
.RS 20n
Read attributes.
.RE

.sp
.ne 2
.na
\fB\fBread_xattr\fR\fR
.ad
.RS 20n
Read named attributes.
.RE

.sp
.ne 2
.na
\fB\fBsynchronize\fR\fR
.ad
.RS 20n
Synchronize.
.RE

.sp
.ne 2
.na
\fB\fBwrite_acl\fR\fR
.ad
.RS 20n
Write ACL.
.RE

.sp
.ne 2
.na
\fB\fBwrite_attributes\fR\fR
.ad
.RS 20n
Write attributes.
.RE
```

```
.sp
.ne 2
.na
\fB\fBwrite_data\fR\fR
.ad
.RS 20n
Write permission.
.RE

.sp
.ne 2
.na
\fB\fBwrite_owner\fR\fR
.ad
.RS 20n
Write owner.
.RE

.sp
.ne 2
.na
\fB\fBwrite_xattr\fR\fR
.ad
.RS 20n
Write named attributes.
.RE

.sp
.LP
This format allows permissions to be specified as, for example:
\fBread_data\fR/\fBread_xattr\fR/\fBread_attributes\fR.
.sp
.LP
When \fBACL_COMPACT_FMT\fR is specified, the permissions consist of 14 unique
letters.  A hyphen (-) character is used to indicate that the permission at
that position is not specified.
.sp
.ne 2
.na
\fB\fBa\fR\fR
.ad
.RS 5n
read attributes
.RE

.sp
.ne 2
.na
\fB\fBA\fR\fR
.ad
.RS 5n
write attributes
.RE

.sp
.ne 2
.na
\fB\fBc\fR\fR
.ad
.RS 5n
read ACL
.RE

.sp
.ne 2
.na
```

```
\fB\fBC\fR\fR
.ad
.RS 5n
write ACL
.RE

.sp
.ne 2
.na
\fB\fBd\fR\fR
.ad
.RS 5n
delete
.RE

.sp
.ne 2
.na
\fB\fBD\fR\fR
.ad
.RS 5n
delete child
.RE

.sp
.ne 2
.na
\fB\fBo\fR\fR
.ad
.RS 5n
write owner
.RE

.sp
.ne 2
.na
\fB\fBp\fR\fR
.ad
.RS 5n
append
.RE

.sp
.ne 2
.na
\fB\fBr\fR\fR
.ad
.RS 5n
read_data
.RE

.sp
.ne 2
.na
\fB\fBR\fR\fR
.ad
.RS 5n
read named attributes
.RE

.sp
.ne 2
.na
\fB\fBs\fR\fR
.ad
.RS 5n
```

```
synchronize
.RE

.sp
.ne 2
.na
\fB\fBw\fR\fR
.ad
.RS 5n
write_data
.RE

.sp
.ne 2
.na
\fB\fBW\fR\fR
.ad
.RS 5n
write named attributes
.RE

.sp
.ne 2
.na
\fB\fBx\fR\fR
.ad
.RS 5n
execute
.RE

.sp
.LP
This format allows compact permissions to be represented as, for example:
\fBrw--d-a-------\fR
.sp
.LP
The fourth field is optional when \fBACL_COMPACT_FMT\fR is not specified, in
which case the field will be present only when the ACL entry has inheritance
flags set. The following is the list of inheritance flags separated by a slash
(/) character.
.sp
.ne 2
.na
\fB\fBdir_inherit\fR\fR
.ad
.RS 16n
\fBACE_DIRECTORY_INHERIT_ACE\fR
.RE

.sp
.ne 2
.na
\fB\fBfile_inherit\fR\fR
.ad
.RS 16n
\fBACE_FILE_INHERIT_ACE\fR
.RE

.sp
.ne 2
.na
\fB\fBinherit_only\fR\fR
.ad
.RS 16n
\fBACE_INHERIT_ONLY_ACE\fR
.RE
```

```
.sp
.ne 2
.na
\fB\fBno_propagate\fR\fR
.ad
.RS 16n
\fBACE_NO_PROPAGATE_INHERIT_ACE\fR
.RE

.sp
.LP
When \fBACL_COMPACT_FMT\fR is specified the inheritance will always be present
and is represented as positional arguments. A hyphen (-) character is used to
indicate that the inheritance flag at that position is not specified.
.sp
.ne 2
.na
\fB\fBd\fR\fR
.ad
.RS 5n
\fBdir_inherit\fR
.RE

.sp
.ne 2
.na
\fB\fBf\fR\fR
.ad
.RS 5n
\fBfile_inherit\fR
.RE

.sp
.ne 2
.na
\fB\fBF\fR\fR
.ad
.RS 5n
failed access (not currently supported)
.RE

.sp
.ne 2
.na
\fB\fBi\fR\fR
.ad
.RS 5n
\fBinherit_only\fR
.RE

.sp
.ne 2
.na
\fB\fBn\fR\fR
.ad
.RS 5n
\fBno_propagate\fR
.RE

.sp
.ne 2
.na
\fB\fBS\fR\fR
.ad
.RS 5n
```

```
successful access (not currently supported)
.RE

.sp
.LP
The fifth field contains the type of the ACE (\fBallow\fR or \fBdeny\fR):
.sp
.ne 2
.na
\fB\fBallow\fR\fR
.ad
.RS 9n
The mask specified in field three should be allowed.
.RE

.sp
.ne 2
.na
\fB\fBdeny\fR\fR
.ad
.RS 9n
The mask specified in field three should be denied.
.RE

.SH RETURN VALUES
.sp
.LP
Upon successful completion, the \fBacl_totext()\fR function returns a pointer
to a text string. Otherwise, it returns \fINULL\fR.
.sp
.LP
Upon successful completion, the \fBacl_fromtext()\fR function returns 0.
Otherwise, the return value is set to one of the following:
.sp
.ne 2
.na
\fB\fBEACL_FIELD_NOT_BLANK\fR\fR
.ad
.RS 28n
A field that should be blank is not blank.
.RE

.sp
.ne 2
.na
\fB\fBEACL_FLAGS_ERROR\fR\fR
.ad
.RS 28n
An invalid ACL flag was specified.
.RE

.sp
.ne 2
.na
\fB\fBEACL_INHERIT_ERROR\fR\fR
.ad
.RS 28n
An invalid inheritance field was specified.
.RE

.sp
.ne 2
.na
\fB\fBEACL_INVALID_ACCESS_TYPE\fR\fR
.ad
.RS 28n
```

```
An invalid access type was specified.
.RE

.sp
.ne 2
.na
\fB\fBEACL_INVALID_STR\fR\fR
.ad
.RS 28n
The string is \fINULL\fR.
.RE

.sp
.ne 2
.na
\fB\fBEACL_INVALID_USER_GROUP\fR\fR
.ad
.RS 28n
The required user or group name not found.
.RE

.sp
.ne 2
.na
\fB\fBEACL_MISSING_FIELDS\fR\fR
.ad
.RS 28n
The ACL needs more fields to be specified.
.RE

.sp
.ne 2
.na
\fB\fBEACL_PERM_MASK_ERROR\fR\fR
.ad
.RS 28n
The permission mask is invalid.
.RE

.sp
.ne 2
.na
\fB\fBEACL_UNKNOWN_DATA\fR\fR
.ad
.RS 28n
Unknown data was found in the ACL.
.RE

.SH EXAMPLES
.LP
\fBExample 1 \fRExamples of permissions when \fBACL_COMPACT_FMT\fR is not
specified.
.sp
.in +2
.nf
user:joe:read_data/write_data:file_inherit/dir_inherit:allow
.fi
.in -2
.sp

.sp
.in +2
.nf
owner@:read_acl:allow,user:tom:read_data:file_inherit/inherit_only:deny
.fi
.in -2
```

```
.sp

.LP
\fBExample 2 \fRExamples of permissions when \fBACL_COMPACT_FMT\fR is
specified.
.sp
.in +2
.nf
user:joe:rw------------:fd----:allow
.fi
.in -2
.sp

.sp
.in +2
.nf
owner@:----------c---:------allow,user:tom:r-------------:f-i---:deny
.fi
.in -2
.sp

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Committed
_
MT-Level        Safe
.TE

.SH SEE ALSO
.sp
.LP
\fBls\fR(1), \fBtar\fR(1), \fBacl\fR(2), \fBmalloc\fR(3C),
\fBaclfromtext\fR(3SEC), \fBacl\fR(5), \fBattributes\fR(5)
```

```
**********************************************************
    4759 Sat Feb  8 11:08:23 2020
new/usr/src/man/man3sec/aclcheck.3sec
12288 getfacl and setfacl could stand improvement
**********************************************************
'\" te
.\" Copyright (c) 2001, Sun Microsystems, Inc.
.\" 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 ACLCHECK 3SEC "Dec 10, 2001"
.SH NAME
aclcheck \- check the validity of an ACL
.SH SYNOPSIS
.LP
.nf
\fBcc\fR [ \fIflag\fR... ] \fIfile\fR... \fB-lsec\fR [ \fIlibrary\fR... ]
#include <sys/acl.h>

\fBint\fR \fBaclcheck\fR(\fBaclent_t *\fR\fIaclbufp\fR, \fBint\fR \fInentries\fR
.fi

.SH DESCRIPTION
.sp
.LP
The \fBaclcheck()\fR function checks the validity of an \fBACL\fR pointed to by
\fIaclbufp.\fR The \fInentries\fR argument is the number of entries contained
in the buffer. The \fIwhich\fR parameter returns the index of the first entry
that is invalid.
.sp
.LP
The function verifies that an \fBACL\fR pointed to by \fIaclbufp\fR is valid
according to the following rules:
.RS +4
.TP
.ie t \(bu
.el o
There must be exactly one \fBGROUP_OBJ\fR \fBACL\fR entry.
.RE
.RS +4
.TP
.ie t \(bu
.el o
There must be exactly one \fBUSER_OBJ\fR \fBACL\fR entry.
.RE
.RS +4
.TP
.ie t \(bu
.el o
There must be exactly one \fBOTHER_OBJ\fR \fBACL\fR entry.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If there are any \fBGROUP\fR \fBACL\fR entries, then the group \fBID\fR in each
group \fBACL\fR entry must be unique.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If there are any \fBUSER\fR \fBACL\fR entries, then the user \fBID\fR in each
user \fBACL\fR entry must be unique.
.RE
.RS +4
```

```
.TP
.ie t \(bu
.el o
If there are any \fBGROUP\fR or \fBUSER\fR \fBACL\fR entries, then there must
be exactly one \fBCLASS_OBJ\fR (\fBACL\fR mask) entry.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If there are any default \fBACL\fR entries, then the following apply:
.RS +4
.TP
.ie t \(bu
.el o
There must be exactly one default \fBGROUP_OBJ\fR \fBACL\fR entry.
.RE
.RS +4
.TP
.ie t \(bu
.el o
There must be exactly one default \fBOTHER_OBJ\fR \fBACL\fR entry.
.RE
.RS +4
.TP
.ie t \(bu
.el o
There must be exactly one default \fBUSER_OBJ\fR \fBACL\fR entry.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If there are any \fBDEF_GROUP\fR entries, then the group \fBID\fR in each
\fBDEF_GROUP\fR \fBACL\fR entry must be unique.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If there are any \fBDEF_USER\fR entries, then the user \fBID\fR in each
\fBDEF_USER\fR \fBACL\fR entry must be unique.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If there are any \fBDEF_GROUP\fR or \fBDEF_USER\fR entries, then there must be
exactly one \fBDEF_CLASS_OBJ\fR (default \fBACL\fR mask) entry.
.RE
.RE
.RS +4
.TP
.ie t \(bu
.el o
If any of the above rules are violated, then the function fails with
\fBerrno\fR set to \fBEINVAL\fR.
.RE
.SH RETURN VALUES
If the \fBACL\fR is valid, \fBaclcheck()\fR will return \fB0\fR. Otherwise
\fBerrno\fR is set to \fBEINVAL\fR and \fBaclcheck()\fR will return one of the
.sp
.LP
If the \fBACL\fR is valid, \fBalcheck()\fR will return \fB0\fR. Otherwise
\fBerrno\fR is set to \fBEINVAL\fR and return code is set to one of the
following:
```

```
.sp
.ne 2
.na
\fB\fBGRP_ERROR\fR\fR
.ad
.RS 19n
There is more than one \fBGROUP_OBJ\fR or \fBDEF_GROUP_OBJ\fR \fBACL\fR entry.
.RE

.sp
.ne 2
.na
\fB\fBUSER_ERROR\fR\fR
.ad
.RS 19n
There is more than one \fBUSER_OBJ\fR or \fBDEF_USER_OBJ\fR \fBACL\fR entry.
.RE

.sp
.ne 2
.na
\fB\fBCLASS_ERROR\fR\fR
.ad
.RS 19n
There is more than one \fBCLASS_OBJ\fR (\fBACL\fR mask) or \fBDEF_CLASS_OBJ\fR
(default \fBACL\fR mask) entry.
.RE

.sp
.ne 2
.na
\fB\fBOTHER_ERROR\fR\fR
.ad
.RS 19n
There is more than one \fBOTHER_OBJ\fR or \fBDEF_OTHER_OBJ\fR \fBACL\fR entry.
.RE

.sp
.ne 2
.na
\fB\fBDUPLICATE_ERROR\fR\fR
.ad
.RS 19n
Duplicate entries of \fBUSER\fR, \fBGROUP\fR, \fBDEF_USER\fR, or
\fBDEF_GROUP\fR.
.RE

.sp
.ne 2
.na
\fB\fBENTRY_ERROR\fR\fR
.ad
.RS 19n
The entry type is invalid.
.RE

.sp
.ne 2
.na
\fB\fBMISS_ERROR\fR\fR
.ad
.RS 19n
Missing an entry. The \fIwhich\fR parameter returns \fB\(mi1\fR in this case.
.RE

.sp
```

```
.ne 2
.na
\fB\fBMEM_ERROR\fR\fR
.ad
.RS 19n
The system cannot allocate any memory. The \fBwhich\fR parameter returns
\fB\(mi1\fR in this case.
.RE

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Evolving
_
MT-Level        Unsafe
.TE

.SH SEE ALSO
.sp
.LP
\fBacl\fR(2), \fBaclsort\fR(3SEC), \fBattributes\fR(5)
```

```
************************************************************
    5497 Sat Feb  8 11:08:23 2020
new/usr/src/man/man3sec/acltotext.3sec
12288 getfacl and setfacl could stand improvement
************************************************************
'\" te
.\" Copyright (c) 2001, Sun Microsystems, Inc.
.\" 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 ACLTOTEXT 3SEC "Dec 10, 2001"
.SH NAME
acltotext, aclfromtext \- convert internal representation to or from external
representation
.SH SYNOPSIS
.LP
.nf
\fBcc\fR [ \fIflag\fR... ] \fIfile\fR... \fB-lsec\fR [ \fIlibrary\fR... ]
#include <sys/acl.h>

\fBchar *\fR\fBacltotext\fR(\fBaclent_t *\fR\fIaclbufp\fR, \fBint\fR \fIaclcnt\f
.fi

.LP
.nf
\fBaclent_t *\fR\fBaclfromtext\fR(\fBchar *\fR\fIacltextp\fR, \fBint *\fR\fIaclc
.fi

.SH DESCRIPTION
.sp
.LP
The \fBacltotext()\fR function converts an internal \fBACL\fR representation
pointed to by \fIaclbufp\fR into an external \fBACL\fR representation. The
space for the external text string is obtained using \fBmalloc\fR(3C). The
caller is responsible for freeing the space upon completion.
caller is responsible for freeing the space upon completion..
.sp
.LP
The \fBaclfromtext()\fR function converts an external \fBACL\fR representation
pointed to by \fIacltextp\fR into an internal \fBACL\fR representation.  The
space for the list of \fBACL\fR entries is obtained using \fBmalloc\fR(3C). The
caller is responsible for freeing the space upon completion. The \fIaclcnt\fR
argument indicates the number of \fBACL\fR entries found.
.sp
.LP
An external \fBACL\fR representation is defined as follows:
.sp
.LP
<acl_entry>[,<acl_entry>]\|.\|.\|.
.sp
.LP
Each <acl_entry> contains one \fBACL\fR entry. The external representation of
an \fBACL\fR entry contains two or three colon-separated fields. The first
field contains the \fBACL\fR entry tag type. The entry type keywords are
defined as:
.sp
.ne 2
.na
\fB\fBuser\fR\fR
.ad
.RS 17n
This \fBACL\fR entry with no \fBUID\fR specified in the \fBACL\fR entry
\fBID\fR field specifies the access granted to the owner of the object.
Otherwise, this \fBACL\fR entry specifies the access granted to a specific
user-name or user-id number.
.RE
```

```
.sp
.ne 2
.na
\fB\fBgroup\fR\fR
.ad
.RS 17n
This \fBACL\fR entry with no \fBGID\fR specified in the \fBACL\fR entry
\fBID\fR field specifies the access granted to the owning group of the object.
Otherwise, this \fBACL\fR entry specifies the access granted to a specific
group-name or group-id number.
.RE

.sp
.ne 2
.na
\fB\fBother\fR\fR
.ad
.RS 17n
This \fBACL\fR entry specifies the access granted to any user or group that
does not match any other \fBACL\fR entry.
.RE

.sp
.ne 2
.na
\fB\fBmask\fR\fR
.ad
.RS 17n
This \fBACL\fR entry specifies the maximum access granted to user or group
entries.
.RE

.sp
.ne 2
.na
\fB\fBdefault:user\fR\fR
.ad
.RS 17n
This \fBACL\fR entry with no uid specified in the \fBACL\fR entry \fBID\fR
field specifies the default access granted to the owner of the object.
Otherwise, this \fBACL\fR entry specifies the default access granted to a
specific user-name or user-\fBID\fR number.
.RE

.sp
.ne 2
.na
\fB\fBdefault:group\fR\fR
.ad
.RS 17n
This \fBACL\fR entry with no gid specified in the \fBACL\fR entry \fBID\fR
field specifies the default access granted to the owning group of the object.
Otherwise, this \fBACL\fR entry specifies the default access granted to a
specific group-name or group-\fBID\fR number.
.RE

.sp
.ne 2
.na
\fB\fBdefault:other\fR\fR
.ad
.RS 17n
This \fBACL\fR entry specifies the default access for other entry.
.RE
```

```
.sp
.ne 2
.na
\fB\fBdefault:mask\fR\fR
.ad
.RS 17n
This \fBACL\fR entry specifies the default access for mask entry.
.RE

.sp
.LP
The second field contains the \fBACL\fR entry \fBID\fR, as follows:
.sp
.ne 2
.na
\fB\fBuid\fR\fR
.ad
.RS 9n
This field specifies a user-name, or user-\fBID\fR if there is no user-name
associated with the user-\fBID\fR number.
.RE

.sp
.ne 2
.na
\fB\fBgid\fR\fR
.ad
.RS 9n
This field specifies a group-name, or group-\fBID\fR if there is no group-name
associated with the group-\fBID\fR number.
.RE

.sp
.ne 2
.na
\fB\fBempty\fR\fR
.ad
.RS 9n
This field is used by the user and group \fBACL\fR entry types.
.RE

.sp
.LP
The third field contains the following symbolic discretionary access
permissions:
.sp
.ne 2
.na
\fB\fBr\fR\fR
.ad
.RS 9n
read permission
.RE

.sp
.ne 2
.na
\fB\fBw\fR\fR
.ad
.RS 9n
write permission
.RE

.sp
.ne 2
.na
```

```
\fB\fBx\fR\fR
.ad
.RS 9n
execute/search permission
.RE

.sp
.ne 2
.na
\fB\fB\(mi\fR \fR
.ad
.RS 9n
no access
.RE

.SH RETURN VALUES
.sp
.LP
Upon successful completion, the \fBacltotext()\fR function returns a pointer to
a text string. Otherwise, it returns \fBNULL\fR.
.sp
.LP
Upon successful completion, the \fBaclfromtext()\fR function returns a pointer
to a list of \fBACL\fR entries. Otherwise, it returns \fBNULL\fR.
.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Evolving
_
MT-Level        Unsafe
.TE

.SH SEE ALSO
.sp
.LP
\fBacl\fR(2), \fBmalloc\fR(3C), \fBattributes\fR(5)
```

```
**********************************************************
   17758 Sat Feb  8 11:08:23 2020
new/usr/src/man/man5/acl.5
12288 getfacl and setfacl could stand improvement
**********************************************************
'\" te
.\" Copyright (c) 2020 Peter Tribble.
.\" Copyright 2014 Nexenta Systems, Inc.  All rights reserved.
.\" Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved.
.\" 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 ACL 5 "Feb 8, 2020"
.TH ACL 5 "Nov 24, 2014"
.SH NAME
acl \- Access Control Lists
.SH DESCRIPTION
.LP
Access control lists (ACLs) are discretionary access control mechanisms that
grant and deny access to files and directories. Two different ACL models are
supported in this release: POSIX-draft ACLs and NFSv4 ACLs.
supported in the Solaris release: POSIX-draft ACLs and NFSv4 ACLs.
.sp
.LP
The older, POSIX-draft model is supported by the UFS file system. This model is
based on a withdrawn ACL POSIX specification that was never standardized. It
was subsequently withdrawn by the POSIX committee.
.sp
.LP
The other model is based on the standards of the NFSv4 working group and is an
approved standard from the Internet Engineering Task Force (IETF). The ZFS file
system uses the NFSv4 model, and provides richer semantics and finer grained
permission capabilities than the POSIX-draft model.
.SS "POSIX-draft ACLs"
.SS "\fBPOSIX\fR-draft \fBACL\fRs"
.LP
POSIX-draft ACLs provide an alternative security mechanism to basic UNIX file
permissions. Their purpose is to further restrict access
permissions in the Solaris release. Their purpose is to further restrict access
to files and directories or to extend permissions to a particular user. ACLs
can be used to change the permissions for the standard owner, group and other
class bits of a file's mode. ACLs can give additional users and groups access
to the file. A directory can also have a special kind of ACL called a
\fBdefault\fR ACL, which defines ACL entries to be inherited by descendents of
the directory. POSIX-draft ACLs have an ACL entry called \fBmask\fR. The mask
defines the maximum permissions that can be granted to additional user and
group entries. Whenever a file is created or its mode is changed by
\fBchmod\fR(1) or \fBchmod\fR(2), the mask is recomputed. It is recomputed to
be the group permission defined in the mode passed to \fBchmod\fR(2).
.sp
.LP
The POSIX-draft ACL model uses the standard \fBrwx\fR model of traditional UNIX
permissions.
.sp
.LP
An ACL is represented as follows:
.sp
.in +2
.nf
\fIacl_entry\fR[,\fIacl_entry\fR]...
.fi
.in -2
.sp

.sp
.LP
```

```
Each \fIacl_entry\fR contains one ACL entry. An ACL entry is represented by two
or three colon-separated(\fB:\fR) fields.
.sp
.ne 2
.na
\fB\fIuser\fR:[\fIuid\fR]:\fIperms\fR\fR
.ad
.RS 21n
If \fIuid\fR blank, it represents the file owner.
.RE

.sp
.ne 2
.na
\fB\fIgroup\fR:[\fIgid\fR]:\fIperms\fR\fR
.ad
.RS 21n
If \fIgid\fR is blank, it represents the owning group.
.RE

.sp
.ne 2
.na
\fB\fIother\fR:\fIperms\fR\fR
.ad
.RS 21n
Represents the file other class.
.RE

.sp
.ne 2
.na
\fB\fImask\fR:\fIperms\fR\fR
.ad
.RS 21n
Defines the \fBMAX\fR permission to hand out.
.RE

.sp
.LP
For example to give user \fBjoe\fR read and write permissions, the ACL entry is
specified as:
.sp
.in +2
.nf
user:joe:rw-
.fi
.in -2
.sp

.SS "NFSv4 ACLs"
The NFSv4 ACL model is based loosely on the Windows NT ACL model. NFSv4 ACLs
.SS "\fBNFS\fRv4 \fBACL\fRs"
.LP
NFSv4 ACL model is based loosely on the Windows NT ACL model. NFSv4 ACLs
provide a much richer ACL model than POSIX-draft ACLs.
.sp
.LP
The major differences between NFSv4 and POSIX-draft ACLs are as follows:
.RS +4
.TP
.ie t \(bu
.el o
NFSv4 ACLs provide finer grained permissions than the \fBrwx\fR model.
.RE
.RS +4
```

```
.TP
.ie t \(bu
.el o
NFSv4 ACLs allow for both \fBALLOW\fR and \fBDENY\fR entries.
.RE
.RS +4
.TP
.ie t \(bu
.el o
NFSv4 ACLs provide a rich set of inheritance semantics. POSIX ACLs also have
inheritance, but with the NFSv4 model you can control the following inheritance
features:
.RS +4
.TP
.ie t \(bu
.el o
Whether inheritance cascades to both files and directories or only to files or
directories.
.RE
.RS +4
.TP
.ie t \(bu
.el o
In the case of directories, you can indicate whether inheritance is applied to
the directory itself, to just one level of subdirectories, or cascades to all
subdirectories of the directory.
.RE
.RE
.RS +4
.TP
.ie t \(bu
.el o
NFSv4 ACLs provide a mechanism for hooking into a system's audit trail.
Currently, illumos does not support this mechanism.
Currently, Solaris does not support this mechanism.
.RE
.RS +4
.TP
.ie t \(bu
.el o
NFSv4 ACLs enable administrators to specify the order in which ACL entries are
checked. With POSIX-draft ACLs the file system reorders ACL entries into a well
defined, strict access, checking order.
.RE
.sp
.LP
POSIX-draft ACL semantics can be achieved with NFSv4 ACLs. However, only some
NFSv4 ACLs can be translated to equivalent POSIX-draft ACLs.
.sp
.LP
Permissions can be specified in three different \fBchmod\fR ACL formats:
verbose, compact, or positional. The verbose format uses words to indicate that
the permissions are separated with a forward slash (\fB/\fR) character. Compact
format uses the permission letters and positional format uses the permission
letters or the hyphen (\fB-\fR) to identify no permissions.
.sp
.LP
The permissions for verbose mode and their abbreviated form in parentheses for
compact and positional mode are described as follows:
.sp
.ne 2
.na
\fBread_data (\fBr\fR)\fR
.ad
.RS 24n
Permission to read the data of the file
```

```
.RE

.sp
.ne 2
.na
\fBlist_directory (\fBr\fR)\fR
.ad
.RS 24n
Permission to list the contents of a directory.
.RE

.sp
.ne 2
.na
\fBwrite_data (\fBw\fR)\fR
.ad
.RS 24n
Permission to modify a file's data anywhere in the file's offset range. This
includes the ability to grow the file or write to any arbitrary offset.
.RE

.sp
.ne 2
.na
\fBadd_file (\fBw\fR)\fR
.ad
.RS 24n
Permission to add a new file to a directory.
.RE

.sp
.ne 2
.na
\fBappend_data (\fBp\fR)\fR
.ad
.RS 24n
The ability to modify the file's data, but only starting at EOF. Currently,
this permission is not supported.
.RE

.sp
.ne 2
.na
\fBadd_subdirectory (\fBp\fR)\fR
.ad
.RS 24n
Permission to create a subdirectory to a directory.
.RE

.sp
.ne 2
.na
\fBread_xattr (\fBR\fR)\fR
.ad
.RS 24n
The ability to read the extended attributes of a file or do a lookup in the
extended attributes directory.
.RE

.sp
.ne 2
.na
\fBwrite_xattr (\fBW\fR)\fR
.ad
.RS 24n
The ability to create extended attributes or write to the extended attributes
```

```
directory.
.RE

.sp
.ne 2
.na
\fBexecute (\fBx\fR)\fR
.ad
.RS 24n
Permission to execute a file.
.RE

.sp
.ne 2
.na
\fBread_attributes (\fBa\fR)\fR
.ad
.RS 24n
The ability to read basic attributes (non-ACLs) of a file. Basic attributes are
considered to be the stat level attributes. Allowing this access mask bit means
that the entity can execute \fBls\fR(1) and \fBstat\fR(2).
.RE

.sp
.ne 2
.na
\fBwrite_attributes (\fBA\fR)\fR
.ad
.RS 24n
Permission to change the times associated with a file or directory to an
arbitrary value.
.RE

.sp
.ne 2
.na
\fBdelete (\fBd\fR)\fR
.ad
.RS 24n
Permission to delete the file.
.RE

.sp
.ne 2
.na
\fBdelete_child (\fBD\fR)\fR
.ad
.RS 24n
Permission to delete a file within a directory.
.RE

.sp
.ne 2
.na
\fBread_acl (\fBc\fR)\fR
.ad
.RS 24n
Permission to read the ACL.
.RE

.sp
.ne 2
.na
\fBwrite_acl (\fBC\fR)\fR
.ad
.RS 24n
```

```
Permission to write the ACL or the ability to execute \fBchmod\fR(1) or
\fBsetfacl\fR(1).
.RE

.sp
.ne 2
.na
\fBwrite_owner (\fBo\fR)\fR
.ad
.RS 24n
Permission to change the owner or the ability to execute \fBchown\fR(1) or
\fBchgrp\fR(1).
.RE

.sp
.ne 2
.na
\fBsynchronize (\fBs\fR)\fR
.ad
.RS 24n
Permission to access a file locally at the server with synchronous reads and
writes. Currently, this permission is not supported.
.RE

.sp
.LP
The following inheritance flags are supported by NFSv4 ACLs:
.sp
.ne 2
.na
\fBfile_inherit (\fBf\fR)\fR
.ad
.RS 26n
Inherit to all newly created files in a directory.
.RE

.sp
.ne 2
.na
\fBdir_inherit (\fBd\fR)\fR
.ad
.RS 26n
Inherit to all newly created directories in a directory.
.RE

.sp
.ne 2
.na
\fBinherit_only (\fBi\fR)\fR
.ad
.RS 26n
Placed on a directory, but does not apply to the directory itself, only to
newly created files and directories. This flag requires file_inherit
and/or dir_inherit to indicate what to inherit.
.RE

.sp
.ne 2
.na
\fBno_propagate (\fBn\fR)\fR
.ad
.RS 26n
Placed on directories and indicates that ACL entries should only be inherited
one level of the tree. This flag requires file_inherit and/or dir_inherit to
indicate what to inherit.
.RE
```

```
.sp
.ne 2
.na
\fBsuccessful_access (\fBS\fR)\fR
.ad
.RS 26n
Indicates whether an alarm or audit record should be initiated upon successful
accesses. Used with audit/alarm ACE types.
.RE

.sp
.ne 2
.na
\fBfailed_access (\fBF\fR)\fR
.ad
.RS 26n
Indicates whether an alarm or audit record should be initiated when access
fails. Used with audit/alarm ACE types.
.RE

.sp
.ne 2
.na
\fBinherited (\fBI\fR)\fR
.ad
.RS 26n
ACE was inherited.
.RE

.sp
.ne 2
.na
\fB\fB-\fR\fR
.ad
.RS 26n
No permission granted.
.RE

.sp
.LP
An NFSv4 ACL is expressed using the following syntax:
.sp
.in +2
.nf
\fIacl_entry\fR[,\fIacl_entry\fR]...

    owner@:<perms>[:inheritance flags]:<allow|deny>
    group@:<perms>[:inheritance flags]:<allow|deny>
    everyone@:<perms>[:inheritance flags]:<allow|deny>
    user:<username>:<perms>[:inheritance flags]:<allow|deny>
    usersid:<sid string>:<perms>[:inheritance flags]:<allow|deny>
    group:<groupname>:<perms>[:inheritance flags]:<allow|deny>
    groupsid:<sid string>:<perms>[:inheritance flags]:<allow|deny>
    sid:<sid string>:<perms>[:inheritance flags]:<allow|deny>
.fi
.in -2

.sp
.ne 2
.na
\fBowner@\fR
.ad
.RS 10n
File owner
.RE
```

```
.sp
.ne 2
.na
\fBgroup@\fR
.ad
.RS 10n
Group owner
.RE

.sp
.ne 2
.na
\fBuser\fR
.ad
.RS 10n
Permissions for a specific user
.RE

.sp
.ne 2
.na
\fBgroup\fR
.ad
.RS 10n
Permissions for a specific group
.RE

.sp
.LP
Permission and inheritance flags are separated by a \fB/\fR character.
.sp
.LP
ACL specification examples:
.sp
.in +2
.nf
user:fred:read_data/write_data/read_attributes:file_inherit:allow
owner@:read_data:allow,group@:read_data:allow,user:tom:read_data:deny
.fi
.in -2
.sp

.sp
.LP
Using the compact ACL format, permissions are specified by using 14 unique
letters to indicate permissions.
.sp
.LP
Using the positional ACL format, permissions are specified as positional
arguments similar to the \fBls -V\fR format. The hyphen (\fB-\fR), which
indicates that no permission is granted at that position, can be omitted and
only the required letters have to be specified.
.sp
.LP
The letters above are listed in the order they would be specified in positional
notation.
.sp
.LP
With these letters you can specify permissions in the following equivalent
ways.
.sp
.in +2
.nf
user:fred:rw------R------:file_inherit:allow
.fi
```

```
.in -2
.sp

.sp
.LP
Or you can remove the \fB-\fR and scrunch it together.
.sp
.in +2
.nf
user:fred:rwR:file_inherit:allow
.fi
.in -2
.sp

.sp
.LP
The inheritance flags can also be specified in a more compact manner, as
follows:
.sp
.in +2
.nf
user:fred:rwR:f:allow
user:fred:rwR:f------:allow
.fi
.in -2
.sp

.SS "Shell-level API"
Several utilities support the manipulation of ACLs. The following
utilities accommodate both ACL models:
.SS "Shell-level Solaris \fBAPI\fR"
.LP
The Solaris command interface supports the manipulation of ACLs. The following
Solaris utilities accommodate both ACL models:
.sp
.ne 2
.na
\fB\fBchmod\fR\fR
.ad
.RS 12n
The \fBchmod\fR utility has been enhanced to allow for the setting and deleting
of ACLs. This is achieved by extending the symbolic-mode argument to support
ACL manipulation. See \fBchmod\fR(1) for details.
.RE

.sp
.ne 2
.na
\fB\fBcompress\fR\fR
.ad
.RS 12n
When a file is compressed any ACL associated with the original file is
preserved with the compressed file.
.RE

.sp
.ne 2
.na
\fB\fBcp\fR\fR
.ad
.RS 12n
By default, \fBcp\fR ignores ACLs, unless the \fB-p\fR option is specified.
When \fB-p\fR is specified the owner and group id, permission modes,
modification and access times, ACLs, and extended attributes if applicable are
preserved.
.RE
```

```
.sp
.ne 2
.na
\fB\fBcpio\fR\fR
.ad
.RS 12n
ACLs are preserved when the \fB-P\fR option is specified.
.RE

.sp
.ne 2
.na
\fB\fBfind\fR\fR
.ad
.RS 12n
Find locates files with ACLs when the \fB-acl\fR flag is specified.
.RE

.sp
.ne 2
.na
\fB\fBls\fR\fR
.ad
.RS 12n
By default \fBls\fR does not display ACL information. When the \fB-v\fR option
is specified, a file’s ACL is displayed.
.RE

.sp
.ne 2
.na
\fB\fBmv\fR\fR
.ad
.RS 12n
When a file is moved, all attributes are carried along with the renamed file.
When a file is moved across a file system boundary, the ACLs are replicated. If
the ACL information cannot be replicated, the move fails and the source file is
not removed.
.RE

.sp
.ne 2
.na
\fB\fBpack\fR\fR
.ad
.RS 12n
When a file is packed, any ACL associated with the original file is preserved
with the packed file.
.RE

.sp
.ne 2
.na
\fB\fBrcp\fR\fR
.ad
.RS 12n
\fBrcp\fR has been enhanced to support copying. A file’s ACL is only preserved
when the remote host supports ACLs.
.RE

.sp
.ne 2
.na
\fB\fBtar\fR\fR
.ad
```

```
642 .RS 12n
643 ACLs are preserved when the \fB-p\fR option is specified.
644 .RE

646 .sp
647 .ne 2
648 .na
649 \fB\fBunpack\fR\fR
650 .ad
651 .RS 12n
652 When a file with an ACL is unpacked, the unpacked file retains the ACL
653 information.
654 .RE

656 .SS "Application-level API"
659 .SS "Application-level \fBAPI\fR"
660 .LP
657 The primary interfaces required to access file system ACLs at the programmatic
658 level are the \fBacl_get()\fR and \fBacl_set()\fR functions. These functions
659 support both POSIX-draft ACLs and NFSv4 ACLs.
660 .SS "Retrieving a file’s ACL"
663 support both POSIX draft ACLs and NFSv4 ACLs.
664 .SS "Retrieving a file’s \fBACL\fR"
661 .in +2
662 .nf
663 int acl_get(const char *path, int flag, acl_t **aclp);
664 int facl_get(int fd, int flag, acl_t **aclp);
665 .fi
666 .in -2

668 .sp
669 .LP
670 The \fBacl_get\fR(3SEC) and \fBfacl_get\fR(3SEC) functions retrieve an ACL on
674 The \fBacl_get\fR(3SEC) and \fBfacl_get\fR(3SEC) functions retrieves an ACL on
671 a file whose name is given by path or referenced by the open file descriptor
672 fd. The flag argument specifies whether a trivial ACL should be retrieved. When
673 the flag argument equals \fBACL_NO_TRIVIAL\fR only ACLs that are not
677 the flag argument equals \fBACL_NO_TRIVIAL\fR then only ACLs that are not
674 trivial are retrieved. The ACL is returned in the \fBaclp\fR argument.
675 .SS "Freeing ACL structure"
679 .SS "Freeing \fBACL\fR structure"
676 .in +2
677 .nf
678 void acl_free(acl_t *aclp);
682 void acl_free(acl_t *aclp)s;
679 .fi
680 .in -2

682 .sp
683 .LP
684 The \fBacl_free()\fR function frees up memory allocated for the argument
685 \fBaclp\fR.
686 .SS "Setting an ACL on a file"
689 \fBaclp;\fR.
690 .SS "Setting an \fBACL\fR on a file"
687 .in +2
688 .nf
689 int acl_set(const char *path, acl_t *aclp);
690 int facl_set(int fd, acl_t *aclp);
691 .fi
692 .in -2

694 .sp
695 .LP
696 The \fBacl_set\fR(3SEC) and \fBfacl_get\fR(3SEC) functions are used for setting
697 an ACL on a file whose name is given by path or referenced by the open file
```

```
698 descriptor \fBfd\fR. The \fBaclp\fR argument specifies the ACL to set. The
699 \fBacl_set\fR(3SEC) function translates a POSIX-draft ACL into a NFSv4 ACL when
700 the target file system supports NFSv4 ACLs. No translation is performed when
703 \fBacl_set\fR(3SEC) translates an POSIX-draft ACL into a NFSv4 ACL when the
704 target file systems supports NFSv4 ACLs. No translation is performed when
701 trying to set an NFSv4 ACL on a POSIX-draft ACL supported file system.
702 .SS "Determining an ACL’s trivialness"
706 .SS "Determining an \fBACL\fR’s trivialness"
703 .in +2
704 .nf
705 int acl_trivial(const char *path);
706 .fi
707 .in -2

709 .sp
710 .LP
711 The \fBacl_trivial()\fR function is used to determine whether a file has a
712 trivial ACL.
713 .SS "Removing all ACLs from a file"
717 .SS "Removing all \fBACL\fRs from a file"
714 .in +2
715 .nf
716 int acl_strip(const char *path, uid_t uid, gid_t gid, mode_t mode);
717 .fi
718 .in -2

720 .sp
721 .LP
722 The \fBacl_strip()\fR function removes all ACLs from a file and replaces them
723 with a trivial ACL based off of the passed in argument mode. After replacing
724 the ACL the owner and group of the file are set to the values specified in the
725 uid and gid parameters.
726 .SS "Converting ACLs to/from external representation"
730 .SS "Converting \fBACL\fRs to/from external representation"
727 .in +2
728 .nf
729 int acl_fromtext(const char *path, acl_t **aclp);
730 char *acl_totext(acl_t *aclp, int flags);
731 .fi
732 .in -2

734 .sp
735 .LP
736 The \fBacl_totext()\fR function converts an internal ACL representation pointed
737 to by aclp into an external representation. See \fBDESCRIPTION\fR for details
738 about external representation.
739 .sp
740 .LP
741 The \fBacl_fromtext()\fR function converts an external representation into an
745 The \fBacl_fromtext()\fR functions converts and external representation into an
742 internal representation. See \fBDESCRIPTION\fR for details about external
743 representation.
744 .SH EXAMPLES
749 .LP
745 The following examples demonstrate how the API can be used to perform basic
746 operations on ACLs.
747 .LP
748 \fBExample 1 \fRRetrieving and Setting an ACL
749 .sp
750 .LP
751 Use the following to retrieve an ACL and set it on another file:

753 .sp
754 .in +2
755 .nf
756 error = acl_get("file", ACL_NO_TRIVIAL, &aclp);
```

```
if (error == 0 && aclp != NULL) {
.in +8
error = acl_set("file2", aclp);
acl_free(aclp);
.in -8
}
```
______***unchanged_portion_omitted_***
```
\&...
.fi
.in -2

.LP
\fBExample 3 \fRDetermining if a File has a Trivial ACL
.sp
.LP
Use the following to determine if a file has a trivial ACL:

.sp
.in +2
.nf
char *file = "file5";
istrivial = acl_trivial(file);

if (istrivial == 0)
.in +8
printf("file %s has a trivial ACL\en", file);
.in -8
else
.in +8
printf("file %s has a NON-trivial ACL\en", file);
.in -8
\&...
.fi
.in -2

.LP
\fBExample 4 \fRRemoving all ACLs from a File
.sp
.LP
Use the following to remove all ACLs from a file, and set a new mode, owner,
and group:

.sp
.in +2
.nf
error = acl_strip("file", 10, 100, 0644);
\&...
.fi
.in -2

.SH SEE ALSO
.LP
\fBchgrp\fR(1), \fBchmod\fR(1), \fBchown\fR(1), \fBcp\fR(1), \fBcpio\fR(1),
\fBfind\fR(1), \fBls\fR(1), \fBmv\fR(1), \fBtar\fR(1), \fBsetfacl\fR(1),
\fBchmod\fR(2), \fBacl\fR(2), \fBstat\fR(2), \fBacl_get\fR(3SEC),
\fBaclsort\fR(3SEC), \fBacl_fromtext\fR(3SEC), \fBacl_free\fR(3SEC),
\fBacl_strip\fR(3SEC), \fBacl_trivial\fR(3SEC)
```