```
***********************************************************
    4201 Wed Jun 18 15:41:36 2014
new/usr/src/cmd/pfexec/pfexec.c
4577 pfexec's error reporting is (at least sometimes) awful
***********************************************************
/*
 * 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 2014 Gary Mills
 * Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*
 * New implementation of pfexec(1) and all of the profile shells.
 *
 * The algorithm is as follows:
 *      first try to derive the shell's path from getexecname();
 *      note that this requires a *hard* link to the program, so
 *      if we find that we are actually executing pfexec, we start
 *      looking at argv[0].
 *      argv[0] is also our fallback in case getexecname doesn't find it.
 */
#include <sys/param.h>
#include <alloca.h>
#include <errno.h>
#include <locale.h>
#include <priv.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define PFEXEC  "pfexec"
#ifndef TEXT_DOMAIN
#define TEXT_DOMAIN     "SYS_TEST"
#endif

#define RES_PFEXEC      1
#define RES_OK          0
#define RES_FAILURE     -1

/*
 * Return the shellname
 */
int
shellname(const char *name, char buf[MAXPATHLEN])
{
        const char *cmd = strrchr(name, '/');
```

```
        if (cmd == NULL)
                cmd = name;
        else
                cmd++;

        if (strncmp(cmd, "pf", 2) != 0)
                return (RES_FAILURE);

        if (strcmp(cmd, PFEXEC) == 0)
                return (RES_PFEXEC);

        if (strlen(name) >= MAXPATHLEN)
                return (RES_FAILURE);

        if (cmd == name) {
                (void) strlcpy(buf, cmd + 2, MAXPATHLEN);
        } else {
                (void) strncpy(buf, name, cmd - name);
                (void) strcpy(buf + (cmd - name), cmd + 2);
        }
        return (RES_OK);

}
______unchanged_portion_omitted_

int
main(int argc, char **argv)
{
        char *cmd;
        char *pset = NULL;
        char pathbuf[MAXPATHLEN];
        int c;
        priv_set_t *wanted;
        int oflag;

        oflag = getpflags(PRIV_PFEXEC);
        if (setpflags(PRIV_PFEXEC, 1) != 0) {
                (void) fprintf(stderr,
                    gettext("pfexec: unable to set PFEXEC flag: %s\n"),
                    strerror(errno));
                perror("setpflags(PRIV_PFEXEC)");
                exit(1);
        }

        if (*argv[0] == '-')
                cmd = argv[0] + 1;
        else
                cmd = argv[0];

        /* Strip "pf" from argv[0], it confuses some shells. */
        if (strncmp(cmd, "pf", 2) == 0) {
                argv[0] += 2;
                /* argv[0] will need to start with '-' again. */
                if (argv[0][-2] == '-')
                        *argv[0] = '-';
        }

        /* If this fails, we just continue with plan B */
        if (shellname(getexecname(), pathbuf) == RES_OK)
                (void) execv(pathbuf, argv);

        switch (shellname(cmd, pathbuf)) {
        case RES_OK:
                (void) execv(pathbuf, argv);
                (void) fprintf(stderr,
                    gettext("pfexec: unable to execute %s: %s\n"),
                    pathbuf, strerror(errno));
                perror(pathbuf);
                return (1);
        case RES_PFEXEC:
        case RES_FAILURE:
                while ((c = getopt(argc, argv, "P:")) != EOF) {
                        switch (c) {
                        case 'P':
                                if (pset == NULL) {
                                        pset = optarg;
                                        break;
                                }
                                /* FALLTHROUGH */
                        default:
                                usage();
                        }
                }
                argc -= optind;
                argv += optind;
                if (argc < 1)
                        usage();

                if (pset != NULL) {
                        if ((wanted = priv_str_to_set(pset, ",", NULL)) ==
                            NULL) {
                                (void) fprintf(stderr,
                                    gettext("pfexec: error parsing "
                                    "privileges: %s\n"), strerror(errno));
                                exit(EXIT_FAILURE);
                        }
                        wanted = priv_str_to_set(pset, ",", NULL);
                        if (setppriv(PRIV_ON, PRIV_INHERITABLE, wanted) != 0) {
                                (void) fprintf(stderr,
                                    gettext("pfexec: error setting "
                                    "privileges: %s\n"), strerror(errno));
                                    gettext("setppriv(): %s\n"),
                                    strerror(errno));
                                exit(EXIT_FAILURE);
                        }
                        (void) setpflags(PRIV_PFEXEC, oflag);
                }

                (void) execvp(argv[0], argv);
                (void) fprintf(stderr,
                    gettext("pfexec: unable to execute %s: %s\n"),
                    argv[0], strerror(errno));
                perror(argv[0]);
                return (1);
        }
        return (1);
}
```

______unchanged_portion_omitted_