pam_mkhomedir.c

```
/* PAM Make Home Dir module

   This module will create a users home directory if it does not exist
   when the session begins. This allows users to be present in central
   database (such as nis, kerb or ldap) without using a distributed
   file system or pre-creating a large number of directories.

   Here is a sample /etc/pam.d/login file for Debian GNU/Linux
   2.1:

   auth       requisite  pam_securetty.so
   auth       sufficient pam_ldap.so
   auth       required   pam_unix.so
   auth       optional   pam_group.so
   auth       optional   pam_mail.so
   account    requisite  pam_time.so
   account    sufficient pam_ldap.so
   account    required   pam_unix.so
   session    required   pam_mkhomedir.so skel=/etc/skel/ umask=0022
   session    required   pam_unix.so
   session    optional   pam_lastlog.so
   password   required   pam_unix.so

   Released under the GNU LGPL version 2 or later
   Copyright (c) Red Hat, Inc. 2009
   Originally written by Jason Gunthorpe <jgg@debian.org> Feb 1999
   Structure taken from pam_lastlogin by Andrew Morgan
     <morgan@parc.power.net> 1996
 */

#include "config.h"

#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <sys/wait.h>
#include <unistd.h>
#include <pwd.h>
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <syslog.h>
#include <signal.h>

#include <security/pam_modules.h>
#include <security/_pam_macros.h>
#include <security/pam_modutil.h>
#include <security/pam_ext.h>

#include "pam_cc_compat.h"
```

```c
#include "pam_inline.h"
#include "pam_i18n.h"

/* argument parsing */
#define MKHOMEDIR_DEBUG      020    /* be verbose about things */
#define MKHOMEDIR_QUIET      040    /* keep quiet about things */

#define LOGIN_DEFS           "/etc/login.defs"
#define UMASK_DEFAULT        "0022"

struct options_t {
  int ctrl;
  const char *umask;
  const char *skeldir;
};
typedef struct options_t options_t;

static void
_pam_parse (const pam_handle_t *pamh, int flags, int argc, const char **argv,
        options_t *opt)
{
   opt->ctrl = 0;
   opt->umask = NULL;
   opt->skeldir = "/etc/skel";

   /* does the application require quiet? */
   if ((flags & PAM_SILENT) == PAM_SILENT)
      opt->ctrl |= MKHOMEDIR_QUIET;

   /* step through arguments */
   for (; argc-- > 0; ++argv)
   {
      const char *str;

      if (!strcmp(*argv, "silent")) {
     opt->ctrl |= MKHOMEDIR_QUIET;
      } else if (!strcmp(*argv, "debug")) {
         opt->ctrl |= MKHOMEDIR_DEBUG;
      } else if ((str = pam_str_skip_prefix(*argv, "umask=")) != NULL) {
     opt->umask = str;
      } else if ((str = pam_str_skip_prefix(*argv, "skel=")) != NULL) {
     opt->skeldir = str;
      } else {
     pam_syslog(pamh, LOG_ERR, "unknown option: %s", *argv);
      }
   }
}

static char*
_pam_conv_str_umask_to_homemode(const char *umask)
{
   unsigned int m = 0777 & ~strtoul(umask, NULL, 8);
   return pam_asprintf("%#o", m);
```

```
}

/* Do the actual work of creating a home dir */
static int
create_homedir (pam_handle_t *pamh, options_t *opt,
        const char *user, const char *dir)
{
   int retval, child;
   struct sigaction newsa, oldsa;
   char *login_umask = NULL;
   char *login_homemode = NULL;

   /* Mention what is happening, if the notification fails that is OK */
   if (!(opt->ctrl & MKHOMEDIR_QUIET))
      pam_info(pamh, _("Creating directory '%s'."), dir);


   D(("called."));

   if (opt->ctrl & MKHOMEDIR_DEBUG) {
        pam_syslog(pamh, LOG_DEBUG, "Executing mkhomedir_helper.");
   }

   /* fetch UMASK from /etc/login.defs if not in argv */
   if (opt->umask == NULL) {
      login_umask = pam_modutil_search_key(pamh, LOGIN_DEFS, "UMASK");
      login_homemode = pam_modutil_search_key(pamh, LOGIN_DEFS, "HOME_MODE");
      if (login_homemode == NULL) {
         if (login_umask != NULL) {
            login_homemode = _pam_conv_str_umask_to_homemode(login_umask);
         } else {
            login_homemode = _pam_conv_str_umask_to_homemode(UMASK_DEFAULT);
         }
      }
   } else {
      login_homemode = _pam_conv_str_umask_to_homemode(opt->umask);
   }

   /*
    * This code arranges that the demise of the child does not cause
    * the application to receive a signal it is not expecting - which
    * may kill the application or worse.
    */
   memset(&newsa, '\0', sizeof(newsa));
   newsa.sa_handler = SIG_DFL;
   sigaction(SIGCHLD, &newsa, &oldsa);

   /* fork */
   child = fork();
   if (child == 0) {
    static char *envp[] = { NULL };
    const char *args[] = { NULL, NULL, NULL, NULL, NULL, NULL };
```

```
 if (pam_modutil_sanitize_helper_fds(pamh, PAM_MODUTIL_PIPE_FD,
                     PAM_MODUTIL_PIPE_FD,
                     PAM_MODUTIL_PIPE_FD) < 0)
     _exit(PAM_SYSTEM_ERR);

 /* exec the mkhomedir helper */
 args[0] = MKHOMEDIR_HELPER;
 args[1] = user;
 args[2] = opt->umask ? opt->umask : UMASK_DEFAULT;
 args[3] = opt->skeldir;
 args[4] = login_homemode;

 DIAG_PUSH_IGNORE_CAST_QUAL;
 execve(MKHOMEDIR_HELPER, (char **)args, envp);
 DIAG_POP_IGNORE_CAST_QUAL;

 /* should not get here: exit with error */
 D(("helper binary is not available"));
 _exit(PAM_SYSTEM_ERR);
} else if (child > 0) {
 int rc;
 while ((rc=waitpid(child, &retval, 0)) < 0 && errno == EINTR);
 if (rc < 0) {
   pam_syslog(pamh, LOG_ERR, "waitpid failed: %m");
   retval = PAM_SYSTEM_ERR;
 } else if (!WIFEXITED(retval)) {
       pam_syslog(pamh, LOG_ERR, "mkhomedir_helper abnormal exit: %d", retval);
       retval = PAM_SYSTEM_ERR;
     } else {
   retval = WEXITSTATUS(retval);
 }
} else {
 D(("fork failed"));
 pam_syslog(pamh, LOG_ERR, "fork failed: %m");
 retval = PAM_SYSTEM_ERR;
}

sigaction(SIGCHLD, &oldsa, NULL);   /* restore old signal handler */

if (opt->ctrl & MKHOMEDIR_DEBUG) {
     pam_syslog(pamh, LOG_DEBUG, "mkhomedir_helper returned %d", retval);
}

if (retval != PAM_SUCCESS && !(opt->ctrl & MKHOMEDIR_QUIET)) {
 pam_error(pamh, _("Unable to create and initialize directory '%s'."),
       dir);
}

free(login_umask);
free(login_homemode);

D(("returning %d", retval));
return retval;
```

```
}

/* --- authentication management functions (only) --- */

int
pam_sm_open_session (pam_handle_t *pamh, int flags, int argc,
             const char **argv)
{
   int retval;
   options_t opt;
   const void *user;
   const struct passwd *pwd;
   struct stat St;

   /* Parse the flag values */
   _pam_parse(pamh, flags, argc, argv, &opt);

   /* Determine the user name so we can get the home directory */
   retval = pam_get_item(pamh, PAM_USER, &user);
   if (retval != PAM_SUCCESS || user == NULL || *(const char *)user == '\0')
   {
      pam_syslog(pamh, LOG_NOTICE, "Cannot obtain the user name.");
      return PAM_USER_UNKNOWN;
   }

   /* Get the password entry */
   pwd = pam_modutil_getpwnam (pamh, user);
   if (pwd == NULL)
   {
      pam_syslog(pamh, LOG_NOTICE, "User unknown.");
      D(("couldn't identify user %s", (const char *) user));
      return PAM_USER_UNKNOWN;
   }

   /* Stat the home directory, if something exists then we assume it is
      correct and return a success*/
   if (stat(pwd->pw_dir, &St) == 0) {
      if (opt.ctrl & MKHOMEDIR_DEBUG) {
          pam_syslog(pamh, LOG_DEBUG, "Home directory %s already exists.",
              pwd->pw_dir);
      }
      return PAM_SUCCESS;
   }

   return create_homedir(pamh, &opt, user, pwd->pw_dir);
}

/* Ignore */
int pam_sm_close_session (pam_handle_t * pamh UNUSED, int flags UNUSED,
             int argc UNUSED, const char **argv UNUSED)
{
   return PAM_SUCCESS;
}
```