

NAME

pw_copy, **pw_dup**, **pw_edit**, **pw_equal**, **pw_fini**, **pw_init**, **pw_make**, **pw_make_v7**, **pw_mkdb**, **pw_lock**, **pw_scan**, **pw_tempname**, **pw_tmp** - functions for passwd file handling

LIBRARY

System Utilities Library (libutil, -lutil)

SYNOPSIS

#include <pwd.h>

#include <libutil.h>

int

pw_copy(*int ffd, int tfd, const struct passwd *pw, struct passwd *oldpw*);

*struct passwd **

pw_dup(*const struct passwd *pw*);

int

pw_edit(*int nosetuid*);

int

pw_equal(*const struct passwd *pw1, const struct passwd *pw2*);

void

pw_fini(*void*);

int

pw_init(*const char *dir, const char *master*);

void

pw_initpwd(*struct passwd *pw*);

*char **

pw_make(*const struct passwd *pw*);

*char **

pw_make_v7(*const struct passwd *pw*);

int

pw_mkdb(*const char *user*);

int

pw_lock(*void*);

*struct passwd **

pw_scan(*const char *line, int flags*);

*const char **

pw_tempname(*void*);

int

pw_tmp(*int mfd*);

DESCRIPTION

The **pw_copy**() function reads a password file from *ffd* and writes it back out to *tfd* possibly with modifications:

- If *pw* is NULL and *oldpw* is not NULL, then the record represented by *oldpw* will not be copied (corresponding to user deletion).
- If *pw* and *oldpw* are not NULL then the record corresponding to *pw* will be replaced by the record corresponding to *oldpw*.
- If *pw* is set and *oldpw* is NULL then the record corresponding to *pw* will be appended (corresponding to user addition).

The **pw_copy**() function returns -1 in case of failure otherwise 0.

The **pw_dup**() function duplicates the *struct passwd* pointed to by *pw* and returns a pointer to the copy, or NULL in case of failure. The new *struct passwd* is allocated with `malloc(3)`, and it is the caller's responsibility to free it with `free(3)`.

The **pw_edit**() function invokes the command specified by the EDITOR environment variable (or `/usr/bin/vi` if EDITOR is not defined) on a temporary copy of the master password file created by **pw_tmp**(). If the file was modified, **pw_edit**() installs it and regenerates the password database. The **pw_edit**() function returns -1 in case of failure, 0 if the file was not modified, and a non-zero positive number if the file was modified and successfully installed.

The **pw_equal**() function compares two *struct passwd* and returns 0 if they are equal.

The **pw_fini**() function destroy the temporary file created by **pw_tmp**() if any, kills any running instance

of EDITOR executed by **pw_edit()** if any, and closes the lock created by **pw_lock()** if any.

The **pw_init()** initializes the static variable representing the path to a password file. *dir* is the directory where the password file is located. If set to NULL, it will default to */etc*. *master* is the name of the password file. If set to NULL? it will default to *master.passwd*

The **pw_initpwd()** function initializes the *passwd* struct to canonical values. The entire structure is zeroed, then *pw_uid* and *pw_gid* are set to -1, and all string pointers are set to point at an internally-defined zero-length string.

The **pw_make()** function creates a properly formatted BSD *passwd(5)* line from a *struct passwd*, and returns a pointer to the resulting string. The string is allocated with *malloc(3)*, and it is the caller's responsibility to free it with *free(3)*.

The **pw_make_v7()** function creates a properly formatted UNIX V7 *passwd(5)* line from a *struct passwd*, and returns a pointer to the resulting string. The string is allocated with *malloc(3)*, and it is the caller's responsibility to free it with *free(3)*.

The **pw_mkdb()** function regenerates the password database by running *pwd_mkdb(8)*. If *user* only the record corresponding to that user will be updated. The **pw_mkdb()** function returns 0 in case of success and -1 in case of failure.

The **pw_lock()** function locks the master password file. It returns a file descriptor to the master password file on success and -1 on failure.

The **pw_scan()** function is a wrapper around the internal libc function **__pw_scan()**. It scans the master password file for a line corresponding to the *line* provided and return a *struct passwd* if it matched an existing record. In case of failure, it returns NULL. Otherwise, it returns a pointer to a *struct passwd* containing the matching record. The *struct passwd* is allocated with *malloc(3)*, and it is the caller's responsibility to free it with *free(3)*.

The **pw_tempname()** function returns the temporary name of the masterfile created via **pw_tmp()**.

The **pw_tmp()** creates and opens a presumably safe temporary password file. If *mfd* is a file descriptor to an open password file, it will be read and written back to the temporary password file. Otherwise if should be set -1. The **pw_tmp()** returns an open file descriptor to the temporary password file or -1 in case of failure.

HISTORY

The functions for *passwd* file handling first appeared in 4.4BSD.

AUTHORS

Portions of this software were developed for the FreeBSD Project by ThinkSec AS and Network Associates Laboratories, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"), as part of the DARPA CHATS research program.

This manual page was written by Baptiste Daroussin <*bapt@FreeBSD.org*>.