

**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*>.