Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'endpwent.3p' command

$ man endpwent.3p

ENDPWENT(3P) POSIX Programmer's Manual ENDPWENT(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

endpwent, getpwent, setpwent ? user database functions

SYNOPSIS

```
#include <pwd.h>
void endpwent(void);
struct passwd *getpwent(void);
void setpwent(void);
```

DESCRIPTION

These functions shall retrieve information about users.

The getpwent() function shall return a pointer to a structure contain? ing the broken-out fields of an entry in the user database. Each entry in the user database contains a passwd structure. If the user database is not already open, getpwent() shall open it and return a pointer to a passwd structure containing the first entry in the database. There? after, it shall return a pointer to a passwd structure containing the next entry in the user database. Successive calls can be used to search the entire user database.

If an end-of-file or an error is encountered on reading, getpwent() shall return a null pointer.

An implementation that provides extended security controls may impose further implementation-defined restrictions on accessing the user data? base. In particular, the system may deny the existence of some or all of the user database entries associated with users other than the caller.

The setpwent() function shall rewind the user database so that the next getpwent() call returns the first entry, allowing repeated searches.

The endpwent() function shall close the user database.

The setpwent() and endpwent() functions shall not change the setting of errno if successful.

On error, the setpwent() and endpwent() functions shall set errno to indicate the error.

Since no value is returned by the setpwent() and endpwent() functions, an application wishing to check for error situations should set errno to 0, then call the function, then check errno.

These functions need not be thread-safe.

RETURN VALUE

On successful completion, getpwent() shall return a pointer to a passwd structure. On end-of-file, getpwent() shall return a null pointer and shall not change the setting of errno. On error, getpwent() shall re? turn a null pointer and errno shall be set to indicate the error.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to getpwuid(), getpwnam(), or getp? went(). The returned pointer, and pointers within the structure, might also be invalidated if the calling thread is terminated.

ERRORS

These functions may fail if:

EINTR A signal was caught during the operation.

EIO An I/O error has occurred.

In addition, getpwent() and setpwent() may fail if:

EMFILE All file descriptors available to the process are currently open.

ENFILE The maximum allowable number of files is currently open in the system.

The following sections are informative.

EXAMPLES

Searching the User Database

The following example uses the getpwent() function to get successive entries in the user database, returning a pointer to a passwd structure that contains information about each user. The call to endpwent() closes the user database and cleans up.

```
#include <pwd.h>
#include <stdio.h>
void printname(uid_t uid)
{
    struct passwd *pwd;
    setpwent();
    while((pwd = getpwent()) != NULL) {
        if (pwd->pw_uid == uid) {
            printf("name=%s\n",pwd->pw_name);
            break;
        }
    }
    endpwent();
}
```

APPLICATION USAGE

These functions are provided due to their historical usage. Applica? tions should avoid dependencies on fields in the password database, whether the database is a single file, or where in the file system name space the database resides. Applications should use getpwuid() whenever possible because it avoids these dependencies.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

endgrent(), getlogin(), getpwnam(), getpwuid()

The Base Definitions volume of POSIX.1?2017, <pwd.h>

COPYRIGHT