NAME

endutxent, **getutxent**, **getutxid**, **getutxline**, **getutxuser**, **pututxline**, **setutxdb**, **setutxent** - user accounting database functions

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <utmpx.h>

void
endutxent(*void*);

*struct utmpx **
getutxent(*void*);

*struct utmpx **
getutxid(*const struct utmpx *id*);

*struct utmpx **
getutxline(*const struct utmpx *line*);

*struct utmpx **
getutxuser(*const char *user*);

*struct utmpx **
pututxline(*const struct utmpx *utmpx*);

int
setutxdb(*int type*, *const char *file*);

void
setutxent(*void*);

DESCRIPTION

These functions operate on the user accounting database which stores records of various system activities, such as user login and logouts, but also system startups and shutdowns and modifications to the system's clock. The system stores these records in three databases, each having a different purpose:

/var/run/utx.active

Log of currently active user login sessions. This file is similar to the traditional *utmp* file. This file only contains process related entries, such as user login and logout records.

/var/log/utx.lastlogin

Log of last user login entries per user. This file is similar to the traditional *lastlog* file. This file only contains user login records for users who have at least logged in once.

/var/log/utx.log

Log of all entries, sorted by date of addition. This file is similar to the traditional *wtmp* file. This file may contain any type of record described below.

Each entry in these databases is defined by the structure *utmpx* found in the include file <*utmpx.h*>:

```
struct utmpx {
        short         ut_type;    /* Type of entry. */
        struct timeval  ut_tv;      /* Time entry was made. */
        char          ut_id[];    /* Record identifier. */
        pid_t         ut_pid;     /* Process ID. */
        char          ut_user[];  /* User login name. */
        char          ut_line[];  /* Device name. */
        char          ut_host[];  /* Remote hostname. */
};
```

The *ut_type* field indicates the type of the log entry, which can have one of the following values:

EMPTY No valid user accounting information.

BOOT_TIME Identifies time of system boot.

SHUTDOWN_TIME
Identifies time of system shutdown.

OLD_TIME Identifies time when system clock changed.

NEW_TIME Identifies time after system clock changed.

USER_PROCESS Identifies a process.

INIT_PROCESS Identifies a process spawned by the init process.

LOGIN_PROCESS Identifies the session leader of a logged-in user.

DEAD_PROCESS Identifies a session leader who has exited.

Entries of type INIT_PROCESS and LOGIN_PROCESS are not processed by this implementation.

Other fields inside the structure are:

ut_tv The time the event occurred. This field is used for all types of entries, except EMPTY.

ut_id An identifier that is used to refer to the entry. This identifier can be used to remove or replace a login entry by writing a new entry to the database containing the same value for *ut_id*. This field is only applicable to entries of type USER_PROCESS, INIT_PROCESS, LOGIN_PROCESS and DEAD_PROCESS.

ut_pid The process identifier of the session leader of the login session. This field is only applicable to entries of type USER_PROCESS, INIT_PROCESS, LOGIN_PROCESS and DEAD_PROCESS.

ut_user The user login name corresponding with the login session. This field is only applicable to entries of type USER_PROCESS and INIT_PROCESS. For INIT_PROCESS entries this entry typically contains the name of the login process.

ut_line The name of the TTY character device, without the leading */dev/* prefix, corresponding with the device used to facilitate the user login session. If no TTY character device is used, this field is left blank. This field is only applicable to entries of type USER_PROCESS and LOGIN_PROCESS.

ut_host
The network hostname of the remote system, connecting to perform a user login. If the user login session is not performed across a network, this field is left blank. This field is only applicable to entries of type USER_PROCESS.

This implementation guarantees all inapplicable fields are discarded. The *ut_user*, *ut_line* and *ut_host* fields of the structure returned by the library functions are also guaranteed to be null-terminated in this implementation.

The **getutxent**() function can be used to read the next entry from the user accounting database.

The **getutxid**() function searches for the next entry in the database of which the behaviour is based on the

ut_type field of *id*. If *ut_type* has a value of BOOT_TIME, SHUTDOWN_TIME, OLD_TIME or NEW_TIME, it will return the next entry whose *ut_type* has an equal value. If *ut_type* has a value of USER_PROCESS, INIT_PROCESS, LOGIN_PROCESS or DEAD_PROCESS, it will return the next entry whose *ut_type* has one of the previously mentioned values and whose *ut_id* is equal.

The **getutxline**() function searches for the next entry in the database whose *ut_type* has a value of USER_PROCESS or LOGIN_PROCESS and whose *ut_line* is equal to the same field in *line*.

The **getutxuser**() function searches for the next entry in the database whose *ut_type* has a value of USER_PROCESS and whose *ut_user* is equal to *user*.

The previously mentioned functions will automatically try to open the user accounting database if not already done so. The **setutxdb**() and **setutxent**() functions allow the database to be opened manually, causing the offset within the user accounting database to be rewound. The **endutxent**() function closes the database.

The **setutxent**() database always opens the active sessions database. The **setutxdb**() function opens the database identified by *type*, whose value is either UTXDB_ACTIVE, UTXDB_LASTLOGIN or UTXDB_LOG. It will open a custom file with filename *file* instead of the system-default if *file* is not null. Care must be taken that when using a custom filename, *type* still has to match with the actual format, since each database may use its own file format.

The **pututxline**() function writes record *utmpx* to the system-default user accounting databases. The value of *ut_type* determines which databases are modified.

Entries of type SHUTDOWN_TIME, OLD_TIME and NEW_TIME will only be written to */var/log/utx.log*.

Entries of type USER_PROCESS will also be written to */var/run/utx.active* and */var/log/utx.lastlogin*.

Entries of type DEAD_PROCESS will only be written to */var/log/utx.log* and */var/run/utx.active* if a corresponding USER_PROCESS, INIT_PROCESS or LOGIN_PROCESS entry whose *ut_id* is equal has been found in the latter.

In addition, entries of type BOOT_TIME and SHUTDOWN_TIME will cause all existing entries in */var/run/utx.active* to be discarded.

All entries whose type has not been mentioned previously, are discarded by this implementation of **pututxline**(). This implementation also ignores the value of *ut_tv*.

RETURN VALUES

The **getutxent**(), **getutxid**(), **getutxline**(), and **getutxuser**() functions return a pointer to an *utmpx* structure that matches the mentioned constraints on success or NULL when reaching the end-of-file or when an error occurs.

The **pututxline**() function returns a pointer to an *utmpx* structure containing a copy of the structure written to disk upon success. It returns NULL when the provided *utmpx* is invalid, or *ut_type* has a value of DEAD_PROCESS and an entry with an identifier with a value equal to the field *ut_id* was not found; the global variable *errno* is set to indicate the error.

The **setutxdb**() function returns 0 if the user accounting database was opened successfully. Otherwise, -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

In addition to the error conditions described in open(2), fdopen(3), fopen(3), fseek(3), the **pututxline**() function can generate the following errors:

[ESRCH] The value of *ut_type* is DEAD_PROCESS, and the process entry could not be found.

[EINVAL] The value of *ut_type* is not supported by this implementation.

In addition to the error conditions described in fopen(3), the **setutxdb**() function can generate the following errors:

[EINVAL] The *type* argument contains a value not supported by this implementation.

[EFTYPE] The file format is invalid.

SEE ALSO

last(1), write(1), getpid(2), gettimeofday(2), tty(4), ac(8), newsyslog(8), utx(8)

STANDARDS

The **endutxent**(), **getutxent**(), **getutxid**(), **getutxline**() and **setutxent**() functions are expected to conform to IEEE Std 1003.1-2008 ("POSIX.1").

The **pututxline**() function deviates from the standard by writing its records to multiple database files, depending on its *ut_type*. This prevents the need for special utility functions to update the other databases, such as the **updlastlogx**() and **updwtmpx**() functions which are available in other implementations. It also tries to replace DEAD_PROCESS entries in the active sessions database when storing USER_PROCESS entries and no entry with the same value for *ut_id* has been found. The

standard always requires a new entry to be allocated, which could cause an unbounded growth of the database.

The **getutxuser**() and **setutxdb**() functions, the *ut_host* field of the *utmpx* structure and SHUTDOWN_TIME are extensions.

HISTORY

These functions appeared in FreeBSD 9.0. They replaced the <*utmp.h*> interface.

AUTHORS

Ed Schouten <*ed@FreeBSD.org*>