Rocky Enterprise Linux 9.2 Manual Pages on command 'utmpxname.3'

$ man utmpxname.3

GETUTENT(3)   Linux Programmer's Manual   GETUTENT(3)

NAME

    getutent, getutid, getutline, pututline, setutent, endutent, utmpname - access utmp file

    entries

SYNOPSIS

    #include <utmp.h>

    struct utmp *getutent(void);

    struct utmp *getutid(const struct utmp *ut);

    struct utmp *getutline(const struct utmp *ut);

    struct utmp *pututline(const struct utmp *ut);

    void setutent(void);

    void endutent(void);

    int utmpname(const char *file);

DESCRIPTION

    New applications should use the POSIX.1-specified "utmpx" versions of these functions; see

    CONFORMING TO.

    utmpname() sets the name of the utmp-format file for the other utmp functions to access.

    If utmpname() is not used to set the filename before the other functions are used, they

    assume _PATH_UTMP, as defined in <paths.h>.

    setutent() rewinds the file pointer to the beginning of the utmp file. It is generally a

    good idea to call it before any of the other functions.

    endutent() closes the utmp file. It should be called when the user code is done accessing

    the file with the other functions.

getutent() reads a line from the current file position in the utmp file. It returns a pointer to a structure containing the fields of the line. The definition of this struc‐ture is shown in utmp(5).

getutid() searches forward from the current file position in the utmp file based upon ut. If ut->ut_type is one of RUN_LVL, BOOT_TIME, NEW_TIME, or OLD_TIME, getutid() will find the first entry whose ut_type field matches ut->ut_type. If ut->ut_type is one of INIT_PROCESS, LOGIN_PROCESS, USER_PROCESS, or DEAD_PROCESS, getutid() will find the first entry whose ut_id field matches ut->ut_id.

getutline() searches forward from the current file position in the utmp file. It scans entries whose ut_type is USER_PROCESS or LOGIN_PROCESS and returns the first one whose ut_line field matches ut->ut_line.

pututline() writes the utmp structure ut into the utmp file. It uses getutid() to search for the proper place in the file to insert the new entry. If it cannot find an appropri‐ate slot for ut, pututline() will append the new entry to the end of the file.

RETURN VALUE

getutent(), getutid(), and getutline() return a pointer to a struct utmp on success, and NULL on failure (which includes the "record not found" case). This struct utmp is allo‐cated in static storage, and may be overwritten by subsequent calls.

On success pututline() returns ut; on failure, it returns NULL.

utmpname() returns 0 if the new name was successfully stored, or -1 on failure.

In the event of an error, these functions errno set to indicate the cause.

ERRORS

ENOMEM Out of memory.

ESRCH Record not found.

setutent(), pututline(), and the getut*() functions can also fail for the reasons de‐scribed in open(2).

FILES

/var/run/utmp

database of currently logged-in users

/var/log/wtmp

database of past user logins

ATTRIBUTES

For an explanation of the terms used in this section, see attributes(7).

Interface	Attribute	Value
getutent()	Thread safety	MT-Unsafe init race:utent race:utentbuf sig:ALRM timer
getutid(), getutline()	Thread safety	MT-Unsafe init race:utent sig:ALRM timer
pututline()	Thread safety	MT-Unsafe race:utent sig:ALRM timer
setutent(), endutent(), utmpname()	Thread safety	MT-Unsafe race:utent

In the above table, utent in race:utent signifies that if any of the functions setutent(), getutent(), getutid(), getutline(), pututline(), utmpname(), or endutent() are used in parallel in different threads of a program, then data races could occur.

CONFORMING TO

XPG2, SVr4.

In XPG2 and SVID 2 the function pututline() is documented to return void, and that is what it does on many systems (AIX, HP-UX). HP-UX introduces a new function _pututline() with the prototype given above for pututline().

All these functions are obsolete now on non-Linux systems. POSIX.1-2001 and POSIX.1-2008, following SUSv1, does not have any of these functions, but instead uses

```
#include <utmpx.h>

struct utmpx *getutxent(void);
struct utmpx *getutxid(const struct utmpx *);
struct utmpx *getutxline(const struct utmpx *);
struct utmpx *pututxline(const struct utmpx *);
void setutxent(void);
void endutxent(void);
```

These functions are provided by glibc, and perform the same task as their equivalents without the "x", but use struct utmpx, defined on Linux to be the same as struct utmp. For completeness, glibc also provides utmpxname(), although this function is not specified by POSIX.1.

On some other systems, the utmpx structure is a superset of the utmp structure, with addi?tional fields, and larger versions of the existing fields, and parallel files are main?tained, often /var/*/utmpx and /var/*/wtmpx.

Linux glibc on the other hand does not use a parallel utmpx file since its utmp structure is already large enough. The "x" functions listed above are just aliases for their coun?terparts without the "x" (e.g., getutxent() is an alias for getutent()).

NOTES

Glibc notes

The above functions are not thread-safe. Glibc adds reentrant versions

```
#include <utmp.h>

int getutent_r(struct utmp *ubuf, struct utmp **ubufp);

int getutid_r(struct utmp *ut,
         struct utmp *ubuf, struct utmp **ubufp);

int getutline_r(struct utmp *ut,
          struct utmp *ubuf, struct utmp **ubufp);
```

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

getutent_r(), getutid_r(), getutline_r():

    _GNU_SOURCE
    || /* since glibc 2.19: */ _DEFAULT_SOURCE
    || /* glibc <= 2.19: */   _SVID_SOURCE || _BSD_SOURCE

These functions are GNU extensions, analogs of the functions of the same name without the _r suffix. The ubuf argument gives these functions a place to store their result. On success, they return 0, and a pointer to the result is written in *ubufp. On error, these functions return -1. There are no utmpx equivalents of the above functions. (POSIX.1 does not specify such functions.)

EXAMPLES

The following example adds and removes a utmp record, assuming it is run from within a pseudo terminal. For usage in a real application, you should check the return values of getpwuid(3) and ttyname(3).

```c
#include <string.h>

#include <stdlib.h>

#include <pwd.h>

#include <unistd.h>

#include <utmp.h>

#include <time.h>

int
main(int argc, char *argv[])
{
    struct utmp entry;

    system("echo before adding entry:;who");

    entry.ut_type = USER_PROCESS;

    entry.ut_pid = getpid();

    strcpy(entry.ut_line, ttyname(STDIN_FILENO) + strlen("/dev/"));

    /* only correct for ptys named /dev/tty[pqr][0-9a-z] */

    strcpy(entry.ut_id, ttyname(STDIN_FILENO) + strlen("/dev/tty"));

    time(&entry.ut_time);

    strcpy(entry.ut_user, getpwuid(getuid())->pw_name);

    memset(entry.ut_host, 0, UT_HOSTSIZE);

    entry.ut_addr = 0;

    setutent();

    pututline(&entry);

    system("echo after adding entry:;who");

    entry.ut_type = DEAD_PROCESS;

    memset(entry.ut_line, 0, UT_LINESIZE);

    entry.ut_time = 0;

    memset(entry.ut_user, 0, UT_NAMESIZE);

    setutent();

    pututline(&entry);

    system("echo after removing entry:;who");

    endutent();

    exit(EXIT_SUCCESS);
}
```

SEE ALSO

getutmp(3), utmp(5)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the
project, information about reporting bugs, and the latest version of this page, can be
found at https://www.kernel.org/doc/man-pages/.

<center>2020-06-09   GETUTENT(3)</center>