NAME

getutent, getutid, getutline, pututline, setutent, endutent, utmpname – access utmp file entry

SYNOPSIS

```
#include <utmp.h>

struct utmp *getutent(void);
struct utmp *getutid(const struct utmp *id);
struct utmp *getutline(const struct utmp *line);
struct utmp *pututline(const struct utmp *utmp);
void setutent(void);
void endutent(void); int utmpname(const char *file);
```

MT-LEVEL

Unsafe

DESCRIPTION

getutent( ), **getutid( )**, **getutline( )**, and **pututline( )** each return a pointer to a **utmp** structure with the following members:

```
char                ut_user[8];     /* user login name */
char                ut_id[4];       /* /sbin/inittab id */
                                    /* (usually line #) */
char                ut_line[12];    /* device name (console, lnxx) */
short               ut_pid;         /* process id */
short               ut_type;        /* type of entry */
struct exit_status  ut_exit;        /* exit status of a process */
                                    /* marked as DEAD_PROCESS */
time_t              ut_time;        /* time entry was made */
```

The structure exit status includes the following members:

```
short    e_termination;    /* termination status */
short    e_exit;           /* exit status */
```

getutent( ) reads in the next entry from a **utmp**-like file. If the file is not already open, it opens it. If it reaches the end of the file, it fails.

getutid( ) searches forward from the current point in the **utmp** file until it finds an entry with a **ut_type** matching *id*→**ut_type** if the type specified is **RUN_LVL**, **BOOT_TIME**, **OLD_TIME**, or **NEW_TIME**. If the type specified in *id* is **INIT_PROCESS**, **LOGIN_PROCESS**, **USER_PROCESS**, or **DEAD_PROCESS**, then **getutid( )** will return a pointer to the first entry whose type is one of these four and whose **ut_id** field matches *id*→**ut_id**. If the end of file is reached without a match, it fails.

getutline( ) searches forward from the current point in the **utmp** file until it finds an entry of the type **LOGIN_PROCESS** or **ut_line** string matching the *line*→**ut_line** string. If the end of file is reached without a match, it fails.

pututline( ) writes out the supplied **utmp** structure into the **utmp** file. It uses **getutid( )** to search forward for the proper place if it finds that it is not already at the proper place. It is expected that normally the user of **pututline( )** will have searched for the proper entry using one of the these routines. If so, **pututline( )** will not search. If **pututline( )** does not

find a matching slot for the new entry, it will add a new entry to the end of the file. It returns a pointer to the **utmp** structure. When called by a non-root user, **pututline()** invokes a **setuid()** root program to verify and write the entry, since **/etc/utmp** is normally writable only by root. In this event, the *ut_name* field must correspond to the actual user name associated with the process; the *ut_type* field must be either **USER_PROCESS** or **DEAD_PROCESS**; and the *ut_line* field must be a device special file and be writable by the user.

setutent() resets the input stream to the beginning of the file. This reset should be done before each search for a new entry if it is desired that the entire file be examined.

endutent() closes the currently open file.

utmpname() allows the user to change the name of the file examined, from **/var/adm/utmp** to any other file. It is most often expected that this other file will be **/var/adm/wtmp**. If the file does not exist, this will not be apparent until the first attempt to reference the file is made. **utmpname()** does not open the file. It just closes the old file if it is currently open and saves the new file name.

RETURN VALUES

A null pointer is returned upon failure to read, whether for permissions or having reached the end of file, or upon failure to write. If the file name given is longer than 79 characters, **utmpname()** returns 0. Otherwise, it returns 1.

FILES

/var/adm/utmp
/var/adm/wtmp

SEE ALSO

getutxent(3C), **ttyslot**(3C), **utmp**(4)

NOTES

The most current entry is saved in a static structure. Multiple accesses require that it be copied before further accesses are made. On each call to either **getutid()** or **getutline()**, the routine examines the static structure before performing more I/O. If the contents of the static structure match what it is searching for, it looks no further. For this reason, to use **getutline()** to search for multiple occurrences, it would be necessary to zero out the static area after each success, or **getutline()** would just return the same structure over and over again. There is one exception to the rule about emptying the structure before further reads are done. The implicit read done by **pututline()** (if it finds that it is not already at the correct place in the file) will not hurt the contents of the static structure returned by the **getutent()**, **getutid()** or **getutline()** routines, if the user has just modified those contents and passed the pointer back to **pututline()**.

These routines use buffered standard I/O for input, but **pututline()** uses an unbuffered non-standard write to avoid race conditions between processes trying to modify the **utmp** and **wtmp** files.