Red Hat Enterprise Linux Release 9.2 Manual Pages on 'wtmp.5' command

$ man wtmp.5

UTMP(5) Linux Programmer's Manual UTMP(5)

NAME

utmp, wtmp - login records

SYNOPSIS

#include <utmp.h>

DESCRIPTION

The utmp file allows one to discover information about who is currently using the system. There may be more users currently using the system, because not all programs use utmp logging.

Warning: utmp must not be writable by the user class "other", because many system programs (foolishly) depend on its integrity. You risk faked system logfiles and modifications of system files if you leave utmp writable to any user other than the owner and group owner of the file.

The file is a sequence of utmp structures, declared as follows in <utmp.h> (note that this is only one of several definitions around; de?tails depend on the version of libc):

```
/* Values for ut_type field, below */

#define EMPTY         0 /* Record does not contain valid info
                           (formerly known as UT_UNKNOWN on Linux) */

#define RUN_LVL       1 /* Change in system run-level (see
                           init(1)) */

#define BOOT_TIME     2 /* Time of system boot (in ut_tv) */
```

```
#define NEW_TIME      3 /* Time after system clock change
                         (in ut_tv) */
#define OLD_TIME      4 /* Time before system clock change
                         (in ut_tv) */
#define INIT_PROCESS  5 /* Process spawned by init(1) */
#define LOGIN_PROCESS 6 /* Session leader process for user login */
#define USER_PROCESS  7 /* Normal process */
#define DEAD_PROCESS  8 /* Terminated process */
#define ACCOUNTING    9 /* Not implemented */
#define UT_LINESIZE      32
#define UT_NAMESIZE      32
#define UT_HOSTSIZE     256
struct exit_status {              /* Type for ut_exit, below */
    short e_termination;          /* Process termination status */
    short e_exit;                 /* Process exit status */
};
struct utmp {
    short   ut_type;              /* Type of record */
    pid_t   ut_pid;               /* PID of login process */
    char    ut_line[UT_LINESIZE]; /* Device name of tty - "/dev/" */
    char    ut_id[4];             /* Terminal name suffix,
                                     or inittab(5) ID */
    char    ut_user[UT_NAMESIZE]; /* Username */
    char    ut_host[UT_HOSTSIZE]; /* Hostname for remote login, or
                                     kernel version for run-level
                                     messages */
    struct  exit_status ut_exit;  /* Exit status of a process
                                     marked as DEAD_PROCESS; not
                                     used by Linux init(1) */
    /* The ut_session and ut_tv fields must be the same size when
       compiled 32- and 64-bit.  This allows data files and shared
       memory to be shared between 32- and 64-bit applications. */
#if __WORDSIZE == 64 && defined __WORDSIZE_COMPAT32
```

```
    int32_t ut_session;          /* Session ID (getsid(2)),
                                    used for windowing */
    struct {
        int32_t tv_sec;          /* Seconds */
        int32_t tv_usec;         /* Microseconds */
    } ut_tv;                     /* Time entry was made */
#else
     long   ut_session;          /* Session ID */
     struct timeval ut_tv;       /* Time entry was made */
#endif
    int32_t ut_addr_v6[4];       /* Internet address of remote
                                    host; IPv4 address uses
                                    just ut_addr_v6[0] */
    char __unused[20];           /* Reserved for future use */
};
/* Backward compatibility hacks */
#define ut_name ut_user
#ifndef _NO_UT_TIME
#define ut_time ut_tv.tv_sec
#endif
#define ut_xtime ut_tv.tv_sec
#define ut_addr ut_addr_v6[0]
```

This structure gives the name of the special file associated with the user's terminal, the user's login name, and the time of login in the form of time(2). String fields are terminated by a null byte ('\0') if they are shorter than the size of the field.

The first entries ever created result from init(1) processing init?tab(5). Before an entry is processed, though, init(1) cleans up utmp by setting ut_type to DEAD_PROCESS, clearing ut_user, ut_host, and ut_time with null bytes for each record which ut_type is not DEAD_PROCESS or RUN_LVL and where no process with PID ut_pid exists. If no empty record with the needed ut_id can be found, init(1) creates a new one. It sets ut_id from the inittab, ut_pid and ut_time to the current values, and ut_type to INIT_PROCESS.

mingetty(8) (or agetty(8)) locates the entry by the PID, changes

ut_type to LOGIN_PROCESS, changes ut_time, sets ut_line, and waits for

connection to be established. login(1), after a user has been authen?

ticated, changes ut_type to USER_PROCESS, changes ut_time, and sets

ut_host and ut_addr. Depending on mingetty(8) (or agetty(8)) and lo?

gin(1), records may be located by ut_line instead of the preferable

ut_pid.

When init(1) finds that a process has exited, it locates its utmp entry

by ut_pid, sets ut_type to DEAD_PROCESS, and clears ut_user, ut_host,

and ut_time with null bytes.

xterm(1) and other terminal emulators directly create a USER_PROCESS

record and generate the ut_id by using the string that suffix part of

the terminal name (the characters following /dev/[pt]ty). If they find

a DEAD_PROCESS for this ID, they recycle it, otherwise they create a

new entry. If they can, they will mark it as DEAD_PROCESS on exiting

and it is advised that they null ut_line, ut_time, ut_user, and ut_host

as well.

telnetd(8) sets up a LOGIN_PROCESS entry and leaves the rest to lo?

gin(1) as usual. After the telnet session ends, telnetd(8) cleans up

utmp in the described way.

The wtmp file records all logins and logouts. Its format is exactly

like utmp except that a null username indicates a logout on the associ?

ated terminal. Furthermore, the terminal name ~ with username shutdown

or reboot indicates a system shutdown or reboot and the pair of termi?

nal names |/} logs the old/new system time when date(1) changes it.

wtmp is maintained by login(1), init(1), and some versions of getty(8)

(e.g., mingetty(8) or agetty(8)). None of these programs creates the

file, so if it is removed, record-keeping is turned off.

FILES

/var/run/utmp

/var/log/wtmp

CONFORMING TO

POSIX.1 does not specify a utmp structure, but rather one named utmpx, with specifications for the fields ut_type, ut_pid, ut_line, ut_id, ut_user, and ut_tv. POSIX.1 does not specify the lengths of the ut_line and ut_user fields.

Linux defines the utmpx structure to be the same as the utmp structure.

Comparison with historical systems

Linux utmp entries conform neither to v7/BSD nor to System V; they are a mix of the two.

v7/BSD has fewer fields; most importantly it lacks ut_type, which causes native v7/BSD-like programs to display (for example) dead or lo?gin entries. Further, there is no configuration file which allocates slots to sessions. BSD does so because it lacks ut_id fields.

In Linux (as in System V), the ut_id field of a record will never change once it has been set, which reserves that slot without needing a configuration file. Clearing ut_id may result in race conditions lead?ing to corrupted utmp entries and potential security holes. Clearing the abovementioned fields by filling them with null bytes is not re?quired by System V semantics, but makes it possible to run many pro?grams which assume BSD semantics and which do not modify utmp. Linux uses the BSD conventions for line contents, as documented above.

System V has no ut_host or ut_addr_v6 fields.

NOTES

Unlike various other systems, where utmp logging can be disabled by re?moving the file, utmp must always exist on Linux. If you want to dis?able who(1), then do not make utmp world readable.

The file format is machine-dependent, so it is recommended that it be processed only on the machine architecture where it was created.

Note that on biarch platforms, that is, systems which can run both 32-bit and 64-bit applications (x86-64, ppc64, s390x, etc.), ut_tv is the same size in 32-bit mode as in 64-bit mode. The same goes for ut_session and ut_time if they are present. This allows data files and shared memory to be shared between 32-bit and 64-bit applications. This is achieved by changing the type of ut_session to int32_t, and that of ut_tv to a struct with two int32_t fields tv_sec and tv_usec.

Since ut_tv may not be the same as struct timeval, then instead of the call:

```
gettimeofday((struct timeval *) &ut.ut_tv, NULL);
```

the following method of setting this field is recommended:

```
struct utmp ut;
struct timeval tv;
gettimeofday(&tv, NULL);
ut.ut_tv.tv_sec = tv.tv_sec;
ut.ut_tv.tv_usec = tv.tv_usec;
```

SEE ALSO

ac(1), date(1), init(1), last(1), login(1), logname(1), lslogins(1), users(1), utmpdump(1), who(1), getutent(3), getutmp(3), login(3), lo?gout(3), logwtmp(3), updwtmp(3)

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/.