

**NAME**

**parse\_lt, in\_lt, in\_ltm, in\_ltm\_s, in\_lts** - functions for parsing and checking login time periods

**LIBRARY**

System Utilities Library (libutil, -lutil)

**SYNOPSIS**

```
#include <sys/types.h>
#include <time.h>
#include <login_cap.h>
```

*login\_time\_t*

```
parse_lt(const char *str);
```

*int*

```
in_lt(const login_time_t *lt, time_t *ends);
```

*int*

```
in_ltm(const login_time_t *lt, struct tm *t, time_t *ends);
```

*int*

```
in_ltm_s(const login_time_t *lt, struct tm *t, time_t *ends);
```

*int*

```
in_lts(const login_time_t *lt, time_t *ends);
```

**DESCRIPTION**

This set of functions may be used for parsing and checking login and session times against a predefined list of allowed login times as used in `login.conf(5)`.

The format of allowed and disallowed session times specified in the *times.allow* and *times.deny* capability fields in a login class are comprised of a prefix which specifies one or more 2- or 3-character day codes, followed by a start and end time in 24 hour format separated by a hyphen. Day codes may be concatenated together to select specific days, or the special mnemonics "Any" and "All" (for any/all days of the week), "Wk" for any day of the week (excluding Saturdays and Sundays) and "Wd" for any weekend day may be used.

For example, the following time period:

```
MoThFrSa1400-2200
```

is interpreted as Monday, Thursday through Saturday between the hours of 2pm and 10pm.

Wd0600-1800  
 means Saturday and Sunday, between the hours of 6am through 6pm, and  
 Any0400-1600  
 means any day of the week, between 4am and 4pm.

Note that all time periods reference system local time.

The **parse\_lt()** function converts the ASCII representation of a time period into a structure of type *login\_time\_t*. This is defined as:

```
typedef struct login_time
{
    u_short lt_start; /* Start time */
    u_short lt_end;   /* End time */
    u_char  lt_dow;   /* Days of week */
} login_time_t;
```

The *lt\_start* and *lt\_end* fields contain the number of minutes past midnight at which the described period begins and ends. The *lt\_dow* field is a bit field, containing one bit for each day of the week and one bit unused. A series *LTM\_\** macros may be used for testing bits individually and in combination. If no bits are set in this field - i.e., it contains the value *LTM\_NONE* - then the entire period is assumed invalid. This is used as a convention to mark the termination of an array of *login\_time\_t* values. If **parse\_lt()** returns a *login\_time\_t* with *lt\_dow* equal to *LTM\_NONE* then a parsing error was encountered.

The remaining functions provide the ability to test a given *time\_t* or struct *tm* value against a specific time period or array of time periods. The **in\_ltm()** function determines whether the given time described by the struct *tm* passed as the second parameter falls within the period described by the first parameter. A boolean value is returned, indicating whether or not the time specified falls within the period. If the time does fall within the time period, and the third parameter to the function is not NULL, the time at which the period ends relative to the time passed is returned.

The **in\_ltm()** function is similar to **in\_ltm()** except that the first parameter must be a pointer to an array of *login\_time\_t* objects, which is up to LC\_MAXTIMES (64) elements in length, and terminated by an element with its *lt\_dow* field set to *LTM\_NONE*.

The **in\_lt()** and **in\_lts()** functions are equivalent to **in\_ltm()** and **in\_ltm()**, respectively, with the second argument set to the current time as returned by *localtime(3)*.

## RETURN VALUES

The **parse\_lt()** function returns a filled in structure of type *login\_time\_t* containing the parsed time

period. If a parsing error occurs, the `lt_dow` field is set to `LTM_NONE` (i.e., 0).

The `in_ltm()` function returns non-zero if the given time falls within the period described by the `login_time_t` passed as the first parameter.

The `in_ltms()` function returns the index of the first time period found in which the given time falls, or -1 if none of them apply.

#### SEE ALSO

`getcap(3)`, `login_cap(3)`, `login_class(3)`, `login.conf(5)`, `termcap(5)`

#### HISTORY

The functions `parse_lt()`, `in_lt()`, `in_ltm()`, `in_ltms()` and `in_lts()` first appeared in FreeBSD 2.1.5.