【Linux Programming】Day17(3)

Tags	
Date	@June 9, 2022
Summary	Time and Date

【Ch4】The Linux Environment

4.3 Time and Date

Times and handled using a defined type, a `time_t`. This is an integer type intended to be large enough to contain dates and times in seconds. It's defined in the header file `time.h`.

```
#include <time.h>

time_t time(time_t *tloc);
```

We can find the low-level time value by calling the `time` function, which returns the number of seconds since the start of the epoch.(January 1, 1970).

It will also write the returned value to a location pointed to by `tloc`, if this isn't a null pointer.

Because the ANSI C committee didn't specify that the `time_t` type be used to measure time intervals in seconds, they invented a function `difftime` to calculate the difference in seconds between two `time_t` values and return a double.

```
#include <time.h>

double difftime(time_t time1, time_t time2);
```

To present the time and date in a more meaningful way, we need to convert the time value into a recognizable time and date.

The function `gmtime` breaks down a low-level time value into a structure containing more usual fields:

```
#include <time.h>

struct tm *gmtime(const time_t timeval);
```

The `struct tm` is defined to contain at least the following members:

tm Member	Description
`int tm_sec`	Seconds, 0-61
`int tm_min`	Minutes, 0-59
`int tm_hour`	Hours, 0-23
`int tm_mday`	Day in the month, 1-31
`int tm_mon`	Month in the year, 0-11 (January = 0)
`int tm_year`	Years since 1900
`int tm_wday`	Day in the week, 0-6 (Sunday = 0)
`int tm_yday`	Day in the year, 0-365
`int tm_isdst`	Daylight savings in effect

To convert a broken-down `tm` structure into a raw `time_t` value, we can use the function `mktime`:

```
#include <time.h>

time_t mktime(struct tm *timeptr);
```

`mktime` will return -1 if the structure cannot be represented as `time_t` value.

For friendly time, and date output provided by the date program, we can use the functions `asctime` and `ctime`.

```
#include <time.h>

char *asctime(const struct tm *timeptr);
char *ctime(const time_t *timeval);
```

The `asctime` function returns a string that represents the time and date given by the `tm` structure `timeptr`.

```
Sun Jun 9 12:34:56 2007\n\0
```

To gain more control of the exact formatting of time and date strings, Linux system provide the `strftime` function. This is rather like a `sprintf` for dates and times and works in a similar way

```
#include <time.h>

size_t strftime(char *s, size_t maxsize, const char *format, struct tm *timeptr);
```

The `strftime` function formats the time and date represented by the `tm` structure pointed to by `timeptr` and places the result in the string `s`. This string is specified as at least maxsize characters long.

Conversion Specifier	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time
%d	Day of the month, 01-31
%H	Hour, 00-23
%I	Hour in 12-hour clock, 01-12
%j	Day of the year, 001-366
%m	Month of the year, 01-12
%M	Minutes, 00-59
%p	a.m. or p.m.
%S	Seconds, 00-61
%u	Day in the week, 1-7 (1 = Monday)
%U	Week in the year, 01-53 (Sunday is the first day of the week.)
%V	Week in the year, 01-53 (Monday is the first day of the week.)
%w	Day in the week, 0-6 (0 = Sunday)
%x	Date in local format

Continued on next page

Conversion Specifier	Description
%X	Time in local format
%y	Year number less 1900
%Y	Year
%Z	Time zone name
%%	A % character

The usual date as given by the date program corresponds to a format string of:

```
%a %b %d %H:%M:%S %Y
```

To help with reading dates, we can use the `strptime` function, which takes a string representing a date and time and creates a `tm` structure representing the same date and time.

```
#include <time.h>

char *strptime(const char *buf, const char *format, struct tm *timeptr);
```

The function returns the format string that has yet to be consumed.