Time and Date

time()

```
#include <time.h>

time_t time(time_t *calptr);
                                    Returns: value of time if OK, -1 on error
```

- returns the time since the Epoch (00:00:00 UTC, January 1, 1970), measured in seconds (calendar time)
- If *calptr* is non-NULL, the return value is also stored in the memory pointed to by *calptr*.
- On success, the value of time in seconds since the Epoch is returned.
- On error, ((time_t)-1) is returned

gettimeofday()

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

int gettimeofday(struct timeval *tp, void *tzp);
                                                    Returns: 0 always
```

- provides greater resolution (up to a microsecond) than the *time* function.
- stores the current time as measured from the Epoch in the memory pointed to by *tp*.
- *tzp* is often set to NULL.
- timeval definition

```
struct timeval {
        time_t tv_sec;      /* seconds */
        long   tv_usec;     /* microseconds */
};
```

gettimeofday()

- example

```
#include <sys/time.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int main()
{
   struct timeval mytime;

   gettimeofday(&mytime, NULL);
   printf("%ld:%ld\n", mytime.tv_sec, mytime.tv_usec);
   return 0;
}
```

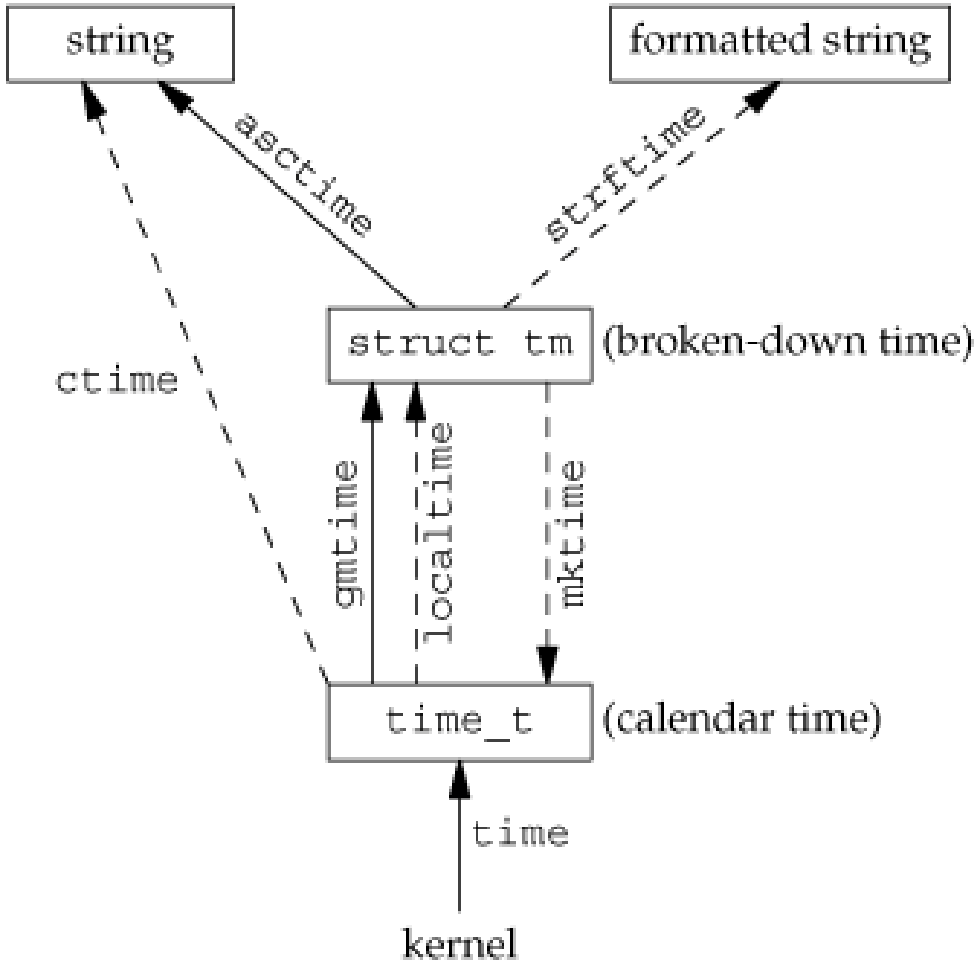
gettimeofday()

- 실 행

```
$ ./a.out
1270980306:878904
$
```

Various time functions

- Relationship of the various time functions



gmtime()/localtime()

- tm structure
  - localtime() and gmtime() convert a calendar time into what's called a broken-down time, a *tm* structure.

```
struct tm {             /* a broken-down time */
   int  tm_sec;         /* seconds after the minute: [0 - 60] */
   int  tm_min;         /* minutes after the hour: [0 - 59] */
   int  tm_hour;        /* hours after midnight: [0 - 23] */
   int  tm_mday;        /* day of the month: [1 - 31] */
   int  tm_mon;         /* months since January: [0 - 11] */
   int  tm_year;        /* years since 1900 */
   int  tm_wday;        /* days since Sunday: [0 - 6] */
   int  tm_yday;        /* days since January 1: [0 - 365] */
   int  tm_isdst;       /* daylight saving time flag: <0, 0, >0 */
 };
```

Positive,	if daylight saving time is in effect
0	if it's not in effect
Negative,	if the information isn't available.

gmtime()/localtime()

```
#include <time.h>

struct tm *gmtime(const time_t *calptr);
struct tm *localtime(const time_t *calptr);
                                        Both return: pointer to broken-down time
```

- gmtime() converts the calendar time into a broken-down time expressed as UTC.
- localtime() converts the calendar time to the local time, taking into account the local time zone and daylight saving time flag.
  - E.g. GMT + 09:00 in Seoul.

mktime()

```
#include <time.h>

time_t mktime(struct tm *tmptr);
                                        Returns: calendar time if OK, -1 on error
```

- takes a broken-down time, expressed as a local time, and c onverts it into a *time_t* value.

asctime()/ctime()

```
#include <time.h>

char *asctime(const struct tm *tmptr);
char *ctime(const time_t *calptr);
                                    Both return: pointer to null-terminated string
```

- Both functions produce the familiar 26-byte string that is similar to the output of the date(1) command:
  - Tue Feb 10 18:27:38 2010\n\0

strftime()

```
#include <time.h>

size_t strftime(char *buf, size_t maxsize, const char *format, const struct tm *tmptr);
                                Returns: number of characters stored in array if room, 0 otherwise
```

- This is the most complicated function for time values.
  - a printf-like function.
- The formatted result is stored in the array *buf* whose size is *maxsize* characters.

strftime()

- Conversion specifiers for strftime

Format	Description	Example
%a	abbreviated weekday name	Tue
%A	full weekday name	Tuesday
%b	abbreviated month name	Feb
%B	full month name	February
%c	date and time	Tue Feb 10 18:27:38 2004
%C	year/100: [00-99]	20
%d	day of the month: [01-31]	10
%D	date [MM/DD/YY]	02/10/04
%e	day of month [1-31]	10
…	…	…

Full descriptions can be referred in your Textbook.

Usage of time related functions

- example

```
#include <stdio.h>
#include <time.h>

int main(void)
{
time_t  t;
char   *ct, buf[80];
struct tm *lt;

        time(&t);
        printf("time:\n\t%ld\n",t);

        ct=ctime(&t);
        printf("ctime:\n\t%s",ct);
```

Usage of time related functions

- example(cont.)

```
        lt=localtime(&t);
        printf("localtime:\n");
        printf("\tyear\t:%d\n", lt->tm_year);
        printf("\tmon\t:%d\n", lt->tm_mon);
        printf("\tday\t:%d\n", lt->tm_mday);
        printf("\thour\t:%d\n", lt->tm_hour);
        printf("\tminute\t:%d\n", lt->tm_min);
        printf("\tsecond\t:%d\n", lt->tm_sec);
        printf("\tweekday\t:%d\n", lt->tm_wday);
        printf("\tyear day\t:%d\n", lt->tm_yday);

        strftime(buf,80,"%A\t%B\t%c",lt);
        printf("strftime:\n\t%s\n", buf);
}
```

Usage of time related functions

- 실 행

```
$ ./a.out
time:
        1270984992
ctime:
        Sun Apr 11 20:23:12 2010
localtime:
        year    :110
        mon     :3
        day     :11
        hour    :20
        minute  :23
        second :12
        weekday :0
        year day:100
strftime:
        Sunday  April   Sun Apr 11 20:23:12 2010
$
```

Usage of time related functions

- Summary through example

Sunday April Sun Apr 11 20:23:12 2010