System Data Files and Information

Time and Date

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

Time and Date

- struct tm

```
{
    int     tm_sec;       /* seconds */
    int     tm_min;        /* minutes */
    int     tm_hour;       /* hours (0~23)*/
    int     tm_mday;        /* day of the month (1~31)*/
    int     tm_mon;        /* month (0~11)*/
    int     tm_year;       /* year (since 1900)*/
    int     tm_wday;        /* day of the week (0~6)*/
    int     tm_yday;       /* day in the year (0~365)*/
    int     tm_isdst;      /* daylight saving time */
};
```

Time and Date

- gmtime( ) and localtime( )
  - struct tm *gmtime(const time_t *timep);
    - function converts the calendar time timep to broken-down time representation, expressed in Coordinated Universal Time (UTC)
  - struct tm *localtime(const time_t *timep);
    - converts the calendar time timep to broken-time representation, expressed relative to the user's specified time zone
    - gmtime() and localtime() uses the same static storage to store the result.

Time and Date

- ctime( ) and asctime( )
  - char *ctime(const time_t *timep);
    - converts the calendar time timep into a string of the form "Wed Jun 30 21:49:08 1993\n”
    - The return value points to a statically allocated string which might be overwritten by subsequent calls to any of the date and time functions
  - char *asctime(const struct tm *timeptr);
    - converts the broken-down time value timeptr into a string with the same format as ctime()

Time and Date

- mktime( )
  - time_t mktime(struct tm *timeptr);
    - converts a broken-down time structure, expressed as local time, to calendar time representation
    - If the specified broken-down time cannot be represented as calendar time returns a value of (time_t)(-1)
- strftime( )
  - size_t strftime(char *s, size_t max, const char *format, const struct tm *tm);

Time and Date

- formats the broken-down time tm according to the format specification format and places the result in the character array s of size max
- %a : The abbreviated weekday name according to the current locale.
- %A : The full weekday name according to the current locale.
- %b : The abbreviated month name according to the current locale.
- %B : The full month name according to the current locale.
- %c : The preferred date and time representation for the current locale.

Time and Date

- %C : The century number (year/100) as a 2-digit integer.
- %d : The day of the month as a decimal number (range 01 to 31)
- %D : Equivalent to %m/%d/%y.
- %e : Like %d, the day of the month as a decimal number, but a leading zero is replaced by a space.
- %h : Equivalent to %b.
- %H : The hour as a decimal number using a 24-hour clock (range 00 to 23).
- %I : The hour as a decimal number using a 12-hour clock (range 01 to 12).

Time and Date

- %j : The day of the year as a decimal number (range 001 to 366).
- %k : The hour (24-hour clock) as a decimal number (range 0 to 23); single digits are preceded by a blank.
- %l : The hour (12-hour clock) as a decimal number (range 1 to 12); single digits are preceded by a blank.
- %m : The month as a decimal number (range 01 to 12).
- %M : The minute as a decimal number (range 00 to 59).
- %n : A newline character.

Time and Date

- %p : Either `AM' or `PM' according to the given time value, or the corresponding strings for the current locale. Noon is treated as `pm' and midnight as `am'.
- %P : Like %p but in lowercase: `am' or `pm' or a corresponding string for the current locale.
- %r : The time in a.m. or p.m. notation. In the POSIX locale this is equivalent to `%I:%M:%S %p'.
- %R : The time in 24-hour notation (%H:%M).
- %s : The number of seconds since the Epoch, i.e., since 1970-01-01 00:00:00 UTC.

Time and Date

- %S : The second as a decimal number (range 00 to 61).
- %t : A tab character.
- %T : The time in 24-hour notation (%H:%M:%S).
- %u : The day of the week as a decimal, range 1 to 7, Monday being 1.
- %U : The week number of the current year as a decimal number, range 00 to 53, starting with the first Sunday as the first day of week 01.
- %w : The day of the week as a decimal, range 0 to 6, Sunday being 0.

Time and Date

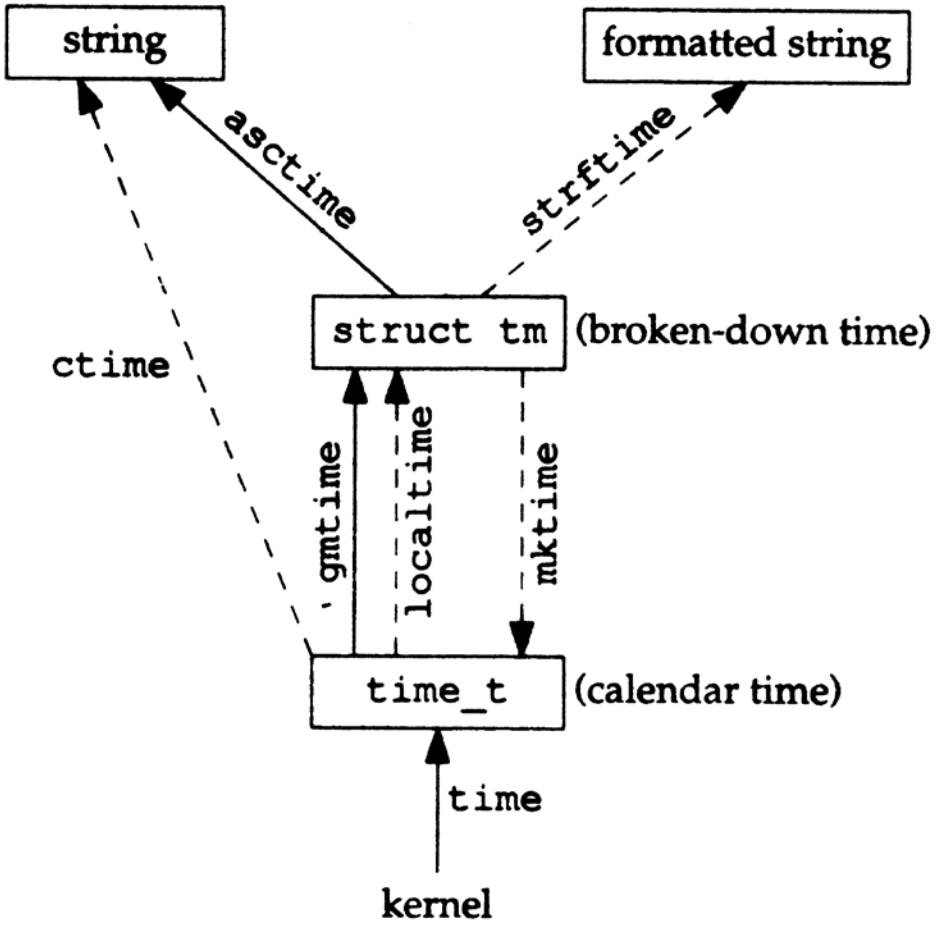
- %W : The week number of the current year as a decimal number, range 00 to 53, starting with the first Monday as the first day of week 01.
- %x : The preferred date representation for the current locale without the time.
- %X : The preferred time representation for the current locale without the date.
- %y : The year as a decimal number without a century (range 00 to 99).
- %Y : The year as a decimal number including the century.

Time and Date

- %z : The time-zone as hour offset from GMT.
- %Z : The time zone or name or abbreviation.
- %+ : The date and time in date( ) format.
- %% : A literal `%' character.

- return value
  - the number of characters placed in the array s, not including the terminating NULL character, 0 on error

Time and Date



Time and Date

- stime()
  - int stime(time_t *t);
    - sets the system's time and date
    - may only be executed by the super user
- gettimeofday()
  - int gettimeofday(struct timeval *tv, struct timezone *tz);

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

Time Example

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

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

        time(&t);
        ct=ctime(&t);
        lt=localtime(&t);
        strftime(buf,80,"%A:%B:%c:%p:%Z",lt);

        printf("time\t: %ld\n",t);
        printf("ctime\t: %s\n",ct);
        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("\tseccond\t:%d\n", lt->tm_sec);
        printf("\tweekday\t:%d\n", lt->tm_wday);
        printf("\tyear day\t:%d\n", lt->tm_yday);
        printf("strftime :%s\n", buf);
}
```