看glibc里面的time 模块code：glibc/time/

头文件：

glibc/include/time.h

glibc/time/time.h

glibc/bits/time.h

UTC时间：

Coordinated Universal Time，世界标准时间，国际协调时间。根据原子钟设定的时间。

各个时区，可以转换成UTC时间，这样就可以相互转换了。

1. **常用结构体定义：**

typedef \_\_time\_t time\_t;

实质是一个long int的数据，表示从1900年到现在的秒钟数。

struct tm

{

int tm\_sec; /\* Seconds. [0-60] (1 leap second) \*/

int tm\_min; /\* Minutes. [0-59] \*/

int tm\_hour; /\* Hours. [0-23] \*/

int tm\_mday; /\* Day. [1-31] \*/

int tm\_mon; /\* Month. [0-11] \*/

int tm\_year; /\* Year - 1900. \*/

int tm\_wday; /\* Day of week. [0-6] \*/

int tm\_yday; /\* Days in year.[0-365] \*/

int tm\_isdst; /\* DST. [-1/0/1]\*/

# ifdef \_\_USE\_MISC

long int tm\_gmtoff; /\* Seconds east of UTC. \*/

const char \*tm\_zone; /\* Timezone abbreviation. \*/

# else

long int \_\_tm\_gmtoff; /\* Seconds east of UTC. \*/

const char \*\_\_tm\_zone; /\* Timezone abbreviation. \*/

# endif

};

tm\_mon，是从0月开始算，所以如果是转换为正常格式，应该要+1。

tm\_year是目前年份和1900的差值，如116，表示2016年(1900+116)。

struct timeval

{

\_\_time\_t tv\_sec; /\* Seconds. \*/

\_\_suseconds\_t tv\_usec; /\* Microseconds. \*/

};

用来表示更精确的时间，tv\_sec表示秒数，tv\_usec表示微秒数。

可以可以用tv\_sec\*1000000 + tv\_usec，获取微秒级别的时间。

stat结构体里的时间

struct stat

{

/\* These are the members that POSIX.1 requires. \*/

\_\_mode\_t st\_mode; /\* File mode. \*/

#ifndef \_\_USE\_FILE\_OFFSET64

\_\_ino\_t st\_ino; /\* File serial number. \*/

#else

\_\_ino64\_t st\_ino; /\* File serial number. \*/

#endif

\_\_dev\_t st\_dev; /\* Device containing the file. \*/

\_\_nlink\_t st\_nlink; /\* Link count. \*/

\_\_uid\_t st\_uid; /\* User ID of the file's owner. \*/

\_\_gid\_t st\_gid; /\* Group ID of the file's group. \*/

#ifndef \_\_USE\_FILE\_OFFSET64

\_\_off\_t st\_size; /\* Size of file, in bytes. \*/

#else

\_\_off64\_t st\_size; /\* Size of file, in bytes. \*/

#endif

\_\_time\_t st\_atime; /\* Time of last access. \*/

\_\_time\_t st\_mtime; /\* Time of last modification. \*/

\_\_time\_t st\_ctime; /\* Time of last status change. \*/

/\* This should be defined if there is a `st\_blksize' member. \*/

#undef \_STATBUF\_ST\_BLKSIZE

};

stat里的st\_mtimei表示文件的修改时间，是time\_t类型数据。

1. **time相关的api**

time\_t time (time\_t \*\_\_timer)

获取当前的时间和1970年1月1日 0时0分0秒的差值，单位为秒，如果\_\_timer不为NULL，则把时间也copy到\_\_timer。

获取的是标准UTC时间，而不是当前时区时间。

double difftime (time\_t \_\_time1, time\_t \_\_time0)

比较2个time\_t数据的差值，表示相差的秒数

struct tm \*localtime (const time\_t \*\_\_timer)

把time\_t类型数据转换为tm类型数据，按照系统设定的时区转换

char \*asctime (const struct tm \*\_\_tp)

/\* Return a string of the form "Day Mon dd hh:mm:ss yyyy\n"

that is the representation of TP in this format. \*/

把tm类型的时间数据转换为特定的格式字符串。

转换 后的是当前时区时间，而不是UTC时间。

char \*ctime (const time\_t \*\_\_timer)

把time\_t类型的时间数据转换为特定的格式字符串。

转换后的是当前时区时间，而不是UTC时间。

int gettimeofday (struct timeval \*\_\_restrict \_\_tv, \_\_timezone\_ptr\_t \_\_tz)

/\* Get the current time of day and timezone information,

putting it into \*TV and \*TZ. If TZ is NULL, \*TZ is not filled.

Returns 0 on success, -1 on errors.

NOTE: This form of timezone information is obsolete.

Use the functions and variables declared in <time.h> instead. \*/

获取系统目前的时间，结构保存在\_\_tv变量里。

获取到的是当前时区的时间，而不是UTC时间。

struct tm \*gmtime (const time\_t \*\_\_timer)

/\* Return the `struct tm' representation of \*TIMER

in Universal Coordinated Time (aka Greenwich Mean Time). \*/

把指定的time\_t数据转换为tm数据格式。gm是GMT的意思，表示格林尼治标准时间。因为GMT时间不准，所以后面都采用的UTC时间，但函数名字没有变化。

转换后的是UTC时间，而非本地时区时间。

gmtime()和localtime()的区别是，gmtime转换后的是UTC时间，而localtime转换后的是本地时区时间，有一个时区时间的转换。

1. **示例**

#include<time.h> //C语言的头文件

#include<stdio.h> //C语言的I/O

void main()

{

time\_t now; //实例化time\_t结构

struct tm \*timenow; //实例化tm结构指针

time(&now);

//time函数读取现在的时间(国际标准时间非北京时间)，然后传值给now

timenow = localtime(&now);

//localtime函数把从time取得的时间now换算成你电脑中的时间(就是你设置的地区)

printf("Local time is %s/n",asctime(timenow));

//上句中asctime函数把时间转换成字符，通过printf()函数输出

}