 UNIX操作系统根据计算机产生的年代和应用采用1970年1月1日作为UNIX的纪元时间，1970年1月1日0点作为计算机表示时间的是中间点，将从1970年1月1日开始经过的秒数用一个整数存放，这种高效简洁的时间表示方法被称为“Unix时间纪元”，向左和向右偏移都可以得到更早或者更后的时间。

在实际开发中，对日期和时间的操作场景非常多，例如程序启动和退出的时间，程序执行任务的时间，数据生成的时间，数据处理的各环节的时间等，无处不在。

在学习时间之前，请把Linux操作系统的时区设置为中国上海时间。

# 一、time\_t别名

在C语言中，用time\_t来表示时间数据类型，它是一个long（长整数）类型的别名，在time.h文件中定义，表示一个日历时间，是从1970年1月1日0时0分0秒到现在的秒数。

typedef long time\_t;

# 二、time库函数

time函数的用途是返回一个值，也就是从1970年1月1日0时0分0秒到现在的秒数。

time函数是C语言标准库中的函数，在time.h文件中声明。

time\_t time(time\_t \*t);

time函数有两种调用方法：

time\_t tnow;

tnow =time(0); // 将空地址传递给time函数，并将time返回值赋给变量tnow

或

time(&tnow); // 将变量tnow的地址作为参数传递给time函数

您可以写代码测试一下这两种方式，效果完全相同。

# 三、tm结构体

time\_t只是一个长整型，不符合我们的使用习惯，需要转换成可以方便表示时间的结构体，即tm结构体，tm结构体在time.h中声明，如下：

struct tm

{

int tm\_sec; // 秒：取值区间为[0,59]

int tm\_min; // 分：取值区间为[0,59]

int tm\_hour; // 时：取值区间为[0,23]

int tm\_mday; // 日期：一个月中的日期：取值区间为[1,31]

int tm\_mon; // 月份：（从一月开始，0代表一月），取值区间为[0,11]

int tm\_year; // 年份：其值等于实际年份减去1900

int tm\_wday; // 星期：取值区间为[0,6]，其中0代表星期天，1代表星期一，以此类推

int tm\_yday; // 从每年的1月1日开始的天数：取值区间为[0,365]，其中0代表1月1日，1代表1月2日，以此类推

int tm\_isdst; // 夏令时标识符，该字段意义不大，我们不用夏令时。

};

这个结构定义了年、月、日、时、分、秒、星期、当年中的某一天、夏令时。用这个结构体可以很方便的显示时间。

# 四、localtime库函数

localtime函数用于把time\_t表示的时间转换为struct tm结构体表示的时间，函数返回struct tm结构体的地址。

函数声明：

 struct tm \* localtime(const time\_t \*);

struct tm结构体包含了时间的各要素，但还不是我们习惯的时间表达方式，我们可以用格式化输出printf、sprintf或fprintf等函数，把struct tm结构体转换为我们想要的结果。

**示例（book128.c）**

/\*

\* 程序名：book128.c，此程序演示获取操作系统时间

\*/

#include <stdio.h>

#include <time.h>

int main(int argc,char \*argv[])

{

time\_t tnow;

tnow=time(0); // 获取当前时间

printf("tnow=%lu\n",tnow); // 输出整数表示的时间

struct tm \*sttm;

sttm=localtime(&tnow); // 把整数的时间转换为struct tm结构体的时间

// yyyy-mm-dd hh24:mi:ss格式输出，此格式用得最多

printf("%04u-%02u-%02u %02u:%02u:%02u\n",sttm->tm\_year+1900,sttm->tm\_mon+1,\

sttm->tm\_mday,sttm->tm\_hour,sttm->tm\_min,sttm->tm\_sec);

printf("%04u年%02u月%02u日%02u时%02u分%02u秒\n",sttm->tm\_year+1900,\

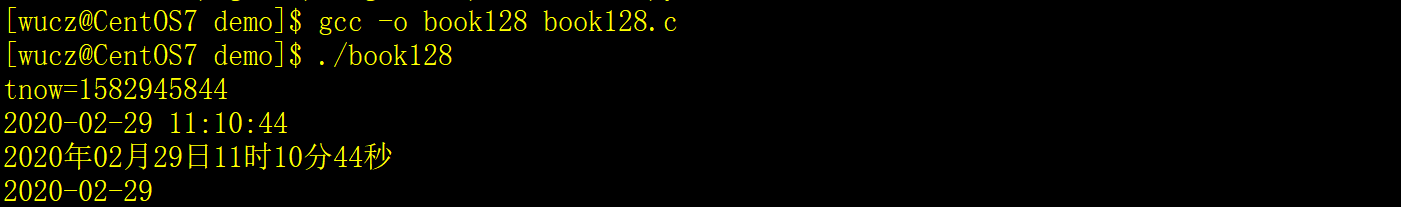
sttm->tm\_mon+1,sttm->tm\_mday,sttm->tm\_hour,sttm->tm\_min,sttm->tm\_sec);

// 只输出年月日

printf("%04u-%02u-%02u\n",sttm->tm\_year+1900,sttm->tm\_mon+1,sttm->tm\_mday);

}

**运行效果**



# 五、mktime库函数

mktime函数的功能与localtime函数相反。

localtime函数用于把time\_t表示的时间转换为struct tm表示的时间。

mktime 函数用于把struct tm表示的时间转换为time\_t表示的时间。

time\_t mktime(struct tm \*tm);

函数返回time\_t的值。

**示例（book130.c）**

/\*

\* 程序名：book130.c，此程序演示时间操作的mktime库函数。

\*/

#include <stdio.h>

#include <time.h>

#include <string.h>

int main(int argc,char \*argv[])

{

// 2019-12-25 15:05:03整数表示是1577257503

struct tm sttm;

memset(&sttm,0,sizeof(sttm));

sttm.tm\_year=2019-1900; // 注意，要减1900

sttm.tm\_mon=12-1; // 注意，要减1

sttm.tm\_mday=25;

sttm.tm\_hour=15;

sttm.tm\_min=5;

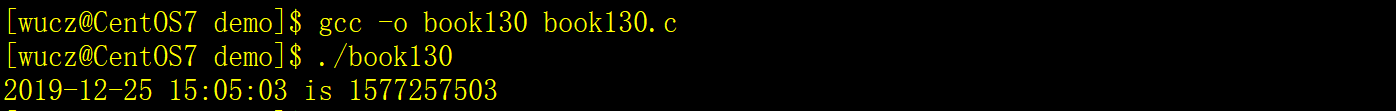
sttm.tm\_sec=3;

sttm.tm\_isdst = 0;

printf("2019-12-25 15:05:03 is %lu\n",mktime(&sttm));

}

**运行效果**



# 六、程序睡眠

在实际开发中，我们经常需要把程序挂起一段时间，可以使用sleep和usleep两个库函数，需要包含unistd.h头文件中。函数的声明如下：

unsigned int sleep(unsigned int seconds);

int usleep(useconds\_t usec);

sleep函数的参数是秒，usleep函数的参数是微秒，1秒=1000000微秒。

sleep(1); // 程序睡眠1秒。

sleep(10); // 程序睡眠10秒。

usleep(100000); // 程序睡眠十分之一秒。

usleep(1000000); // 程序睡眠一秒。

程序员不关心sleep和usleep函数的返回值。

# 七、精确到微秒的计时器

## 1、精确到微秒的timeval结构体

timeval结构体在sys/time.h文件中定义，声明为：

struct timeval

{

long tv\_sec; // 1970年1月1日到现在的秒。

long tv\_usec; // 当前秒的微妙，即百万分之一秒。

};

## 2、时区timezone 结构体

timezone 结构体在sys/time.h文件中定义，声明为：

struct timezone

{

int tz\_minuteswest; // 和UTC（格林威治时间）差了多少分钟。

int tz\_dsttime; // type of DST correction，修正参数据，忽略

};

## 3、gettimeofday库函数

gettimeofday是获得当前的秒和微秒的时间，其中的秒是指1970年1月1日到现在的秒，微秒是指当前秒已逝去的微秒数，可以用于程序的计时。调用gettimeofday函数需要包含sys/time.h头文件。

函数声明：

int gettimeofday(struct  timeval \*tv, struct  timezone \*tz )

当前的时间存放在tv 结构体中，当地时区的信息则放到tz所指的结构体中，tz可以为空。

函数执行成功后返回0，失败后返回-1。

在使用gettimeofday()函数时，第二个参数一般都为空，我们一般都只是为了获得当前时间，不关心时区的信息。

**示例（book132.c）**

/\*

\* 程序名：book132.c，此程序演示精确到微秒的计时器。

\*/

#include <stdio.h>

#include <sys/time.h> // 注意，不是time.h

int main()

{

struct timeval begin,end; // 定义用于存放开始和结束的时间

gettimeofday(&begin,0); // 计时器开始

printf("begin time(0)=%d,tv\_sec=%d,tv\_usec=%d\n",time(0),begin.tv\_sec,begin.tv\_usec);

sleep(2);

usleep(100000); // 程序睡眠十分之一秒。

gettimeofday(&end,0); // 计时器结束

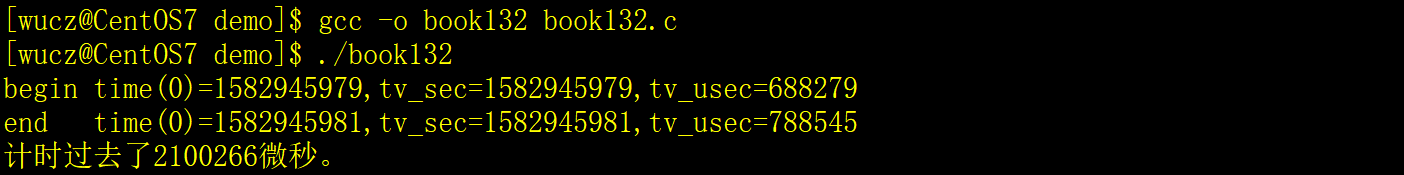
printf("end time(0)=%d,tv\_sec=%d,tv\_usec=%d\n",time(0),end.tv\_sec,end.tv\_usec);

printf("计时过去了%d微秒。\n",\

(end.tv\_sec-begin.tv\_sec)\*1000000+(end.tv\_usec-begin.tv\_usec));

}

**运行效果**



各位，book132.c程序采用usleep睡眠十分之一秒，但是计时器显示的实际时间大于十分之一秒，为何？原因很简单，因为程序执行需要时间，虽然这个时间很短，在千分之一秒内，那也是需要时间。

还有一个要注意的问题，time.h 是ISO C99 标准日期时间头文件。sys/time.h 是Linux 系统的日期时间头文件，也就是说，timeval、timezone结构体和gettimeofday函数在windows平台中不能使用，真是麻烦。