《Linux系统编程》第08期
时间管理和定时器编程

时间的概念

- 计算机中处处需要时间
  - 程序运行时间(音视频播放进度、下载、上传)
  - 系统日志log
  - QQ、微信、短信聊天记录
  - 当前时间、上班时间、睡觉时间
  - 周期性做一些事情(杀毒、清理垃圾、软件升级更新)
  - 定时开关机
  - OS调度：时间片、定时器

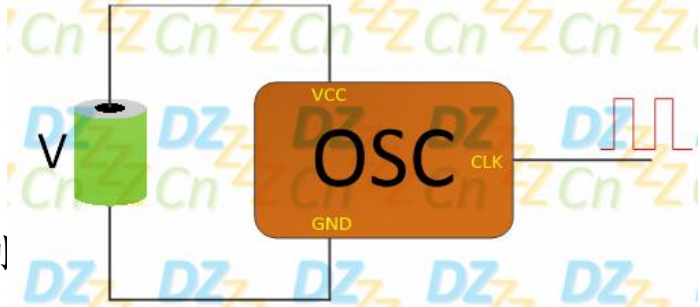
时间的概念

- 时间分类
  - 绝对时间：2020年1月5日、开机时间
  - 相对时间：万历15年、明天、下周、明年、昨天
- 时区的概念
  - 标准时间：
    - UTC：Coordinated Universal Time，世界协调时间
    - GMT：Greenwich Mean Time，格林威治标准时间、子午线
  - 本地时间：
    - 北京时间：东8区、UTC/GMT+08:00
    - 计时误差：手表时间快了？慢了？

时间的计量

时间的计量

- 计时器的发展历程
  - 日出日落、月圆月缺、春夏交替
  - 日晷、沙漏、刻漏
  - 机械钟
  - 石英振荡器、晶振
  - 铯原子钟
  - 氢原子钟、铷原子钟



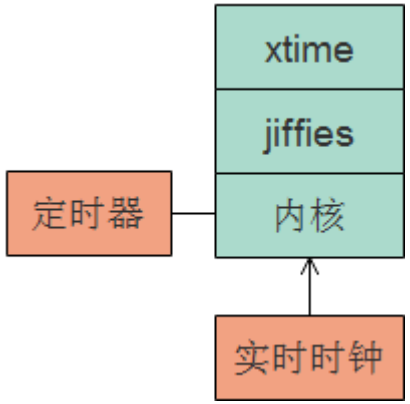
时间的计量

- 计算机中的计时
  - 51单片机：晶振
  - 嵌入式系统：定时器
  - PC计算机：定时器
  - 实时时钟：独立电源(CMOS电池、手机晶振)
- 时间之源
  - 原子时钟
    - 铯原子钟、铷原子钟
    - 各地实验室原子钟
    - 中国科学院国家授时中心
    - 系统时间校准

Linux系统中的时间管理

Linux系统中的时间管理

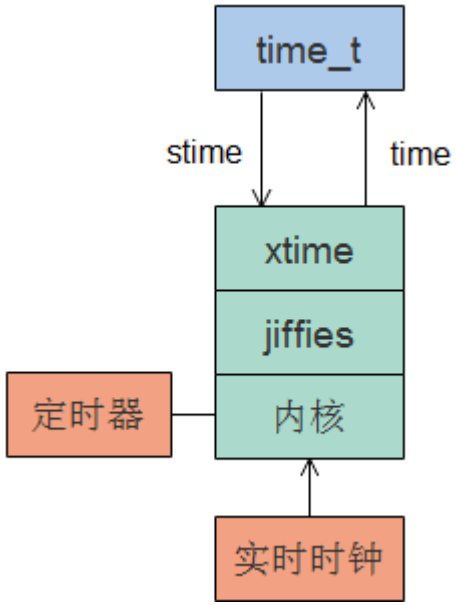
- Linux内核对时间的维护
  - 定时器—时钟中断--时钟节拍tick—jiffies
  - 实时时钟--xtime
  - jiffies：内核中的全局变量，系统启动以来的节拍数
  - HZ：中断频率
  - xtime：内核中的全局变量，墙上时间、UTC



获取当前时间

获取当前时间

- 全局变量：xtime
  - typedef long time_t;
  - time_t time(time_t *tloc);
  - int stime(const time_t *t);

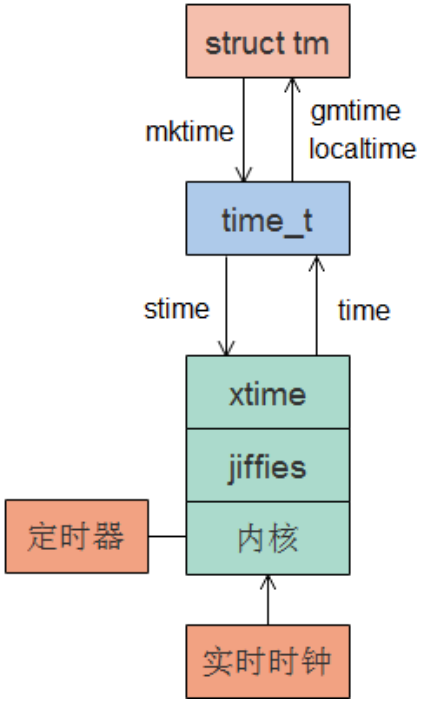


时间格式转换

时间格式转换

- 把秒数转换为年月日时分秒
  - struct tm *gmtime (const time_t *timep);
  - struct tm *localtime (const time_t *timep);
  - time_t mktime(struct tm *tm);

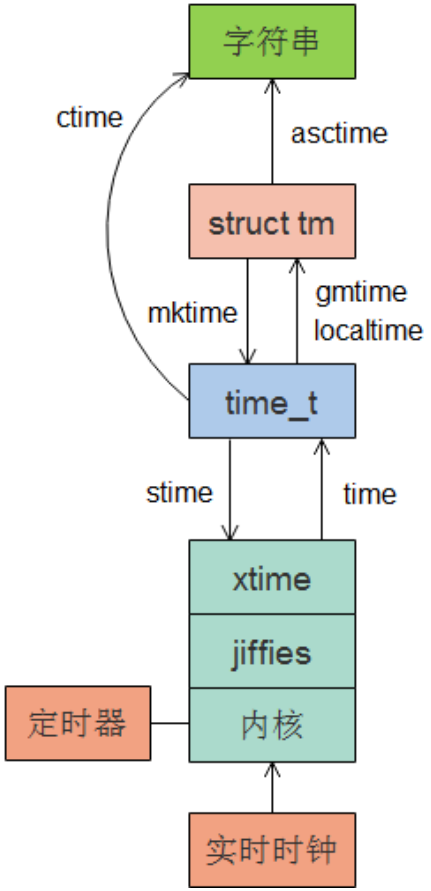
```
struct tm
{
        int tm_sec;     /* Seconds (0-60) */
        int tm_min;     /* Minutes (0-59) */
        int tm_hour;    /* Hours (0-23) */
        int tm_mday;    /* Day of the month (1-31) */
        int tm_mon;     /* Month (0-11) */
        int tm_year;    /* Year - 1900 */
        int tm_wday;    /* Day of the week (0-6, Sunday = 0) */
        int tm_yday;    /* Day in the year (0-365, 1 Jan = 0) */
        int tm_isdst;   /* Daylight saving time */
};
```



将时间转换为字符串

将时间转换为字符串

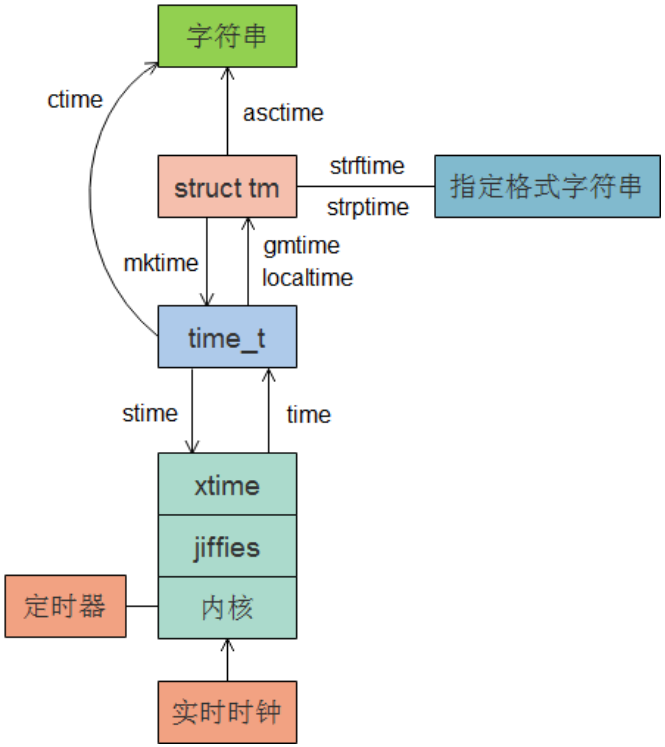
- 时间格式->字符串格式
  - char *asctime(const struct tm *tm);
  - char *ctime(const time_t *timep);



用户指定格式的字符串

指定格式的字符串

- 用户自定义格式
  - size_t strftime (char *s, size_t max, const char *format, const struct tm *tm);
  - char *strptime (const char *s, const char *format, struct tm *tm);



指定格式的字符串

- 字符串格式

%a： 星期几名称的简写
%A： 星期几名称的全称
%b： 月份名称的简写
%B： 月份名称的全称
%c： 首选的日期和时间表示法
%C： 世纪（年份除以 100， 00 到 99）
%d： 一个月中的第几天（01 到 31）
%D： 时间格式，与 %m/%d/%y 表示法相同
%： 一个月中的第几天（1 到 31）
%g： 与 %G 表示法类似，但不带世纪
%h： 与 %b 表示法相同
%H - 小时，使用 24 小时制（00 到 23）
%I： 小时，使用 12 小时制（01 到 12）
%j： 一年中的第几天（001 到 366）
%m： 月份（01 到 12）
%M： 分

%n： 换行符
%p： 与给定的时间值相对应的 am 或 pm
%r： a.m. 和 p.m. 的时间标记法
%R： 24 小时制的时间标记法
%S： 秒
%t： tab 制表符
%T： 当前时间，与 %H:%M:%S 表示法相同
%u： 星期几的数字表示（1 到 7）
%U： 当年周数
%w： 十进制数形式表示一周中的某天，Sunday[星期日] = 0
%x： 首选的日期表示法，不带时间
%X： 首选的时间表示法，不带日期%y
%Y： 包含表示世纪的数字的年份表示
%Z： 时区名称
%z： 时区名称的简写
%%： 输出一个 % 字符

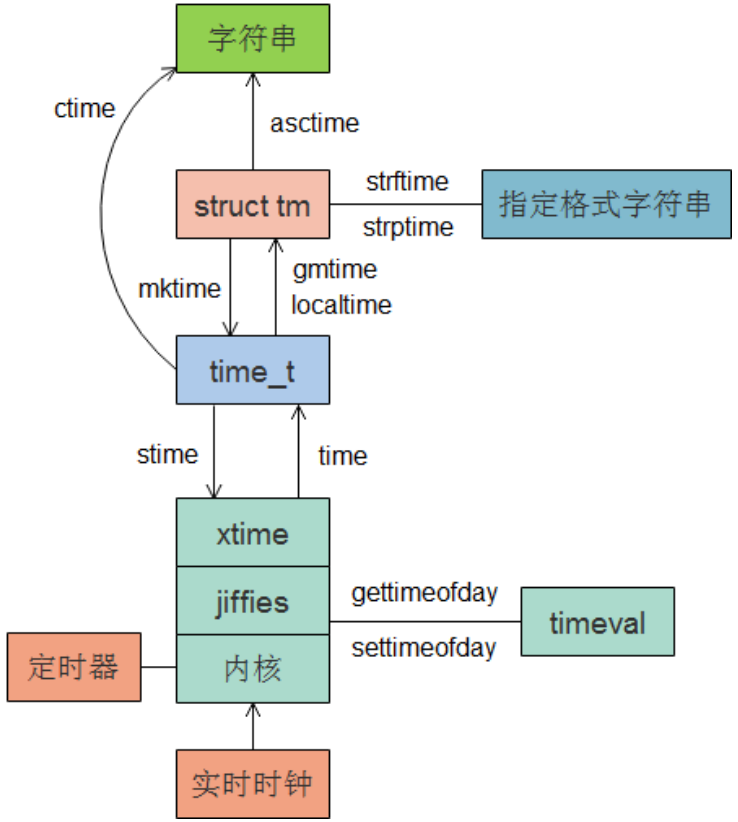
获取高精度时间：微秒

获取高精度时间

- 微秒级时间
  - int gettimeofday (struct timeval *tv, struct timezone *tz);
  - int settimeofday (const struct timeval *tv, const struct timezone *tz);

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

```
struct timezone
{
   int tz_minuteswest;    /* minutes west of Greenwich */
   int tz_dsttime;        /* type of DST correction */
};
```

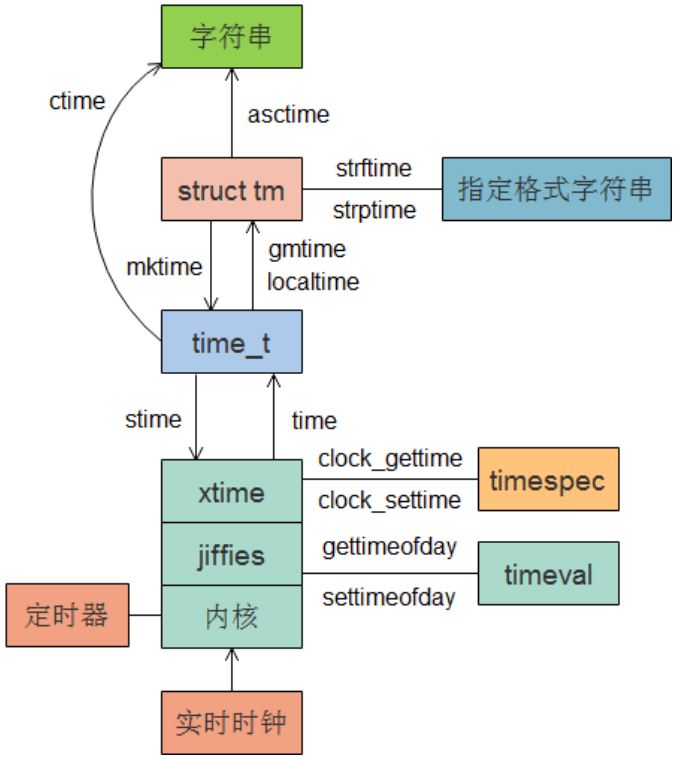


获取高精度时间：纳秒

获取高精度时间

- 纳秒级时间
  - int clock_gettime (clockid_t clk_id, struct timespec *tp);
  - int clock_settime (clockid_t clk_id, const struct timespec *tp);

```
struct timespec
{
   time_t   tv_sec;      /* seconds */
   long     tv_nsec;     /* nanoseconds */
 };
```



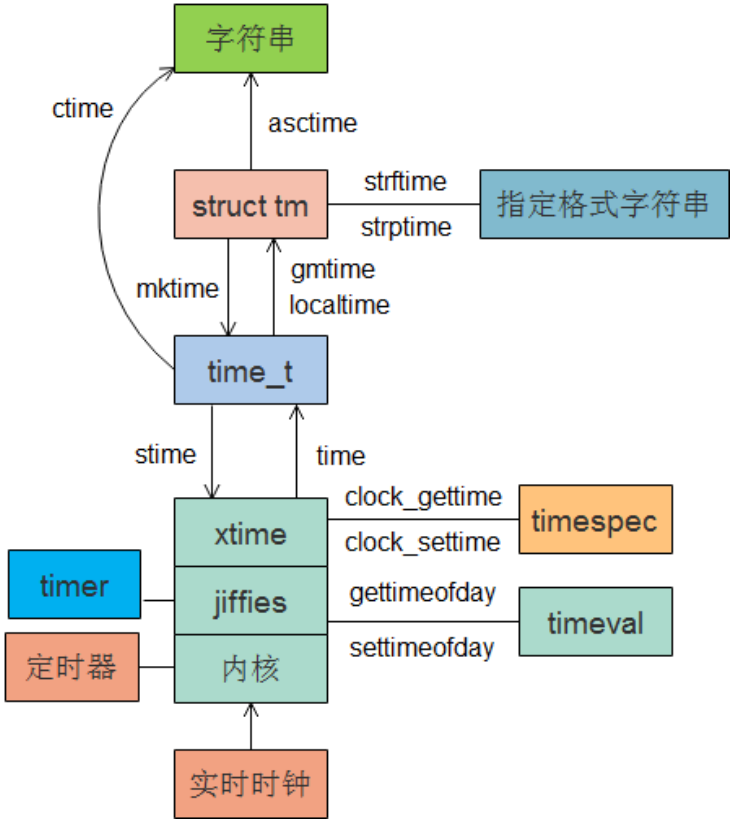
获得高精度时间

- 纳秒级时间
  - 参数：clock_id
    - 说明：设置时钟时间类型
    - CLOCK_REALTIME：系统实时时间(从1970-1-1 计时)
    - CLOCK_MONOTONIC：系统启动时间
    - CLOCK_PROCESS_CPUTIME_ID：当前进程CPU花费时间
    - CLOCK_THREAD_CPUTIME_ID：当前线程CPU花费时间

Linux中的定时器

Linux中的定时器

- 内核对定时器的维护
  - 定时器中断—OS维护全局变量—系统调用API—获取/设置时间
  - 定时器中断— 周期动作：发信号、新建线程
  - 周期性工作：
    - 定时杀毒
    - 数据备份
    - 邮件提醒
    - 日志清理



Linux中的定时器

- 相关API
  - 内核API：add_timer、mod_timer、del_timer
  - 简单的闹钟：alarm、sleep、usleep、nanosleep
  - 计时器(interval timer)：getitimer、setitimer
  - 高级定时器(POSIX timer)
    - timer_create
    - timer_settime
    - timer_gettimer
    - timer_delete
    - timer_getoverrun

Linux中的定时器：interval timer

Linux中的定时器

- 定时器+signal
  - interval timer，共享相同的定时器
    - 定时器到期，发送信号到当前进程【定时器精度：微秒级】
    - int getitimer (int which, struct itimerval *curr_value);
    - int setitimer (int which, const struct itimerval *new_value,
      struct itimerval *old_value);

```
struct itimerval
{
    struct timeval it_interval; /* Interval for periodic timer */
    struct timeval it_value;    /* Time until next expiration */
};

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

Linux中的定时器

- 函数参数
  - which：定时器类型
    - ITIMER_REAL：实时计时，到期后发送SIGALRM信号到进程
    - ITIMER_VIRTUAL：进程用户态运行才计时，到期发送SIGVTALRM信号
    - ITIMER_PROF：进程在用户和内核空间都计时，到期发送SIGPROF信号

Linux中的定时器：POSIX timer

Linux中的定时器

- POSIX定时器API
  - 编译链接：-lrt
  - int timer_create (clockid_t clockid, struct sigevent *sevp, timer_t *timerid);
  - int timer_gettime (timer_t timerid, struct itimerspec *curr_value);
  - int timer_settime (timer_t timerid, int flags, const struct itimerspec *new_value,
    struct itimerspec *old_value);
  - int timer_getoverrun (timer_t timerid);
  - int timer_delete (timer_t timerid);

Linux中的定时器

- POSIX定时器
  - 函数参数： sevp

```
typedef struct sigevent
 {
        __sigval_t    sigev_value;     // timer的ID
        int           sigev_signo;     // 发送信号类型
        int           sigev_notify;    // 定时器到期之后的动作：发信号、线程
        void (*sigev_notify_function) (__sigval_t);   /* Function to start.  */
        void *sigev_notify_attributes;                /* Really pthread_attr_t.*/
 } sigevent_t;
```

```
union sigval
{
    int     sival_int;     // args
    void    *sival_ptr;    //timer ID
};
```

```
struct    itimerspec
 {
        struct timespec      it_interval;
        struct timespec      it_value;
 };
```

Linux中的定时器

- POSIX定时器
  - 函数参数：clockid
    - CLOCK_REALTIME：系统实时时间
    - CLOCK_MONOTONIC：系统启动以来的绝对时间
    - CLOCK_PROCESS_CPUTIME_ID：进程运行时间(用户和内核空间)
    - CLOCK_THREAD_CPUTIME_ID：线程运行时间(用户和内核空间)
  - sevp.sigev_notify：定时器到期后的动作
    - SIGEV_NONE：什么都不做
    - SIGEV_SIGNAL：发送信号给当前进程
    - SIGEV_THREAD：开启一个线程，执行用户指定function
    - SIGEV_THREAD_ID：给当前进程中的指定线程发信号