口令文件

口令文件存储位置: /etc/passwd

struct passwd *getpwuid(uid_t uid)

struct passwd *getpwnam(const char *name)

成功，返回指针，失败，返回NULL, 返回的结果是该类函数维护的一个静态变量，两次调用函数就会对其指向地址的内容产生重写

```
1 struct passwd
2 {
3     char *pw_name;   //用户名
4     char *pw_passwd;//用户口令
5     uid_t pw_uid;   //用户id
6     gid_t pw_gid;   //用户组id
7     char *pw_gecos; //注释字段
8     char *pw_dir;    //初始工作目录
9     char *pw_shell; //初始shell
10    char *pw_class; //用户访问类
11    time_t pw_change;//下次更改口令日期
12    time_t pw_expire;   //账户有效期时间
13 };
```

struct passwd *getpwent(void)

返回值，若成功，返回下一个用户项，出错或者到达文件末尾，返回NULL

void setpwent(void)

将指向用户项的文件指针指向文件首部

void endpwent(void)

关闭这个文件

阴影口令

口令文件存储位置：/etc/shadow

为了防止非管理员用户得到用户口令的加密后的口令，然后用试探来得到明文，就将用户的加密口令存放在阴影文件中，阴影文件只允许root用户进行读写。

struct spwd *getspname(const char *name)

struct spwd*getspent(void)

void setspent(void)

void endspent(void)

```
15 struct spwd
16 {
17     char *sp_namp; //用户登录名
18     char *sp_pwdp; //加密口令
19     int sp_lstchg; //上次口令更改至今的时间
20     int sp_min;    //多少天后允许更改
21     int sp_max;    //要求更改的剩余天数
22     int sp_wan;    //超期警告天数
23     int sp_inact; //账号不活动之前剩余天数
24     int sp_expire; //超期天数
25     unsigned int sp_flag; //保留
26 };
```

其他数据文件

组文件

口令文件

阴影文件

主机文件

网络文件

协议文件

服务文件

…

系统提供了一些操作函数接口，用来结构化获取对应文件中的信息

时间和日期

日历时间:

从协调时间1970/01/01/00:00:00开始到现在的秒数目

精度为秒， 使用time_t 结构来表示

也被称为实时系统时间

操作：

1. 获取日历时间
time_t time(time_t *time) //失败，返回-1

2.获取指定时钟时间
int clock_gettime(clockid_t clock_id, struct timespec *tsp)
//成功，返回0,失败,返回-1

3.获取指定时钟系统可以提供的精度
int clock_getres(clockid_t clock_id, struct timespec *tsp)
//成功，返回 0 ，出错，返回-1

4.设置特定时钟的时间
int clock_settime(clockid_t clock_id, const struct timespec *tsp)
//成功，返回0,失败，返回-1

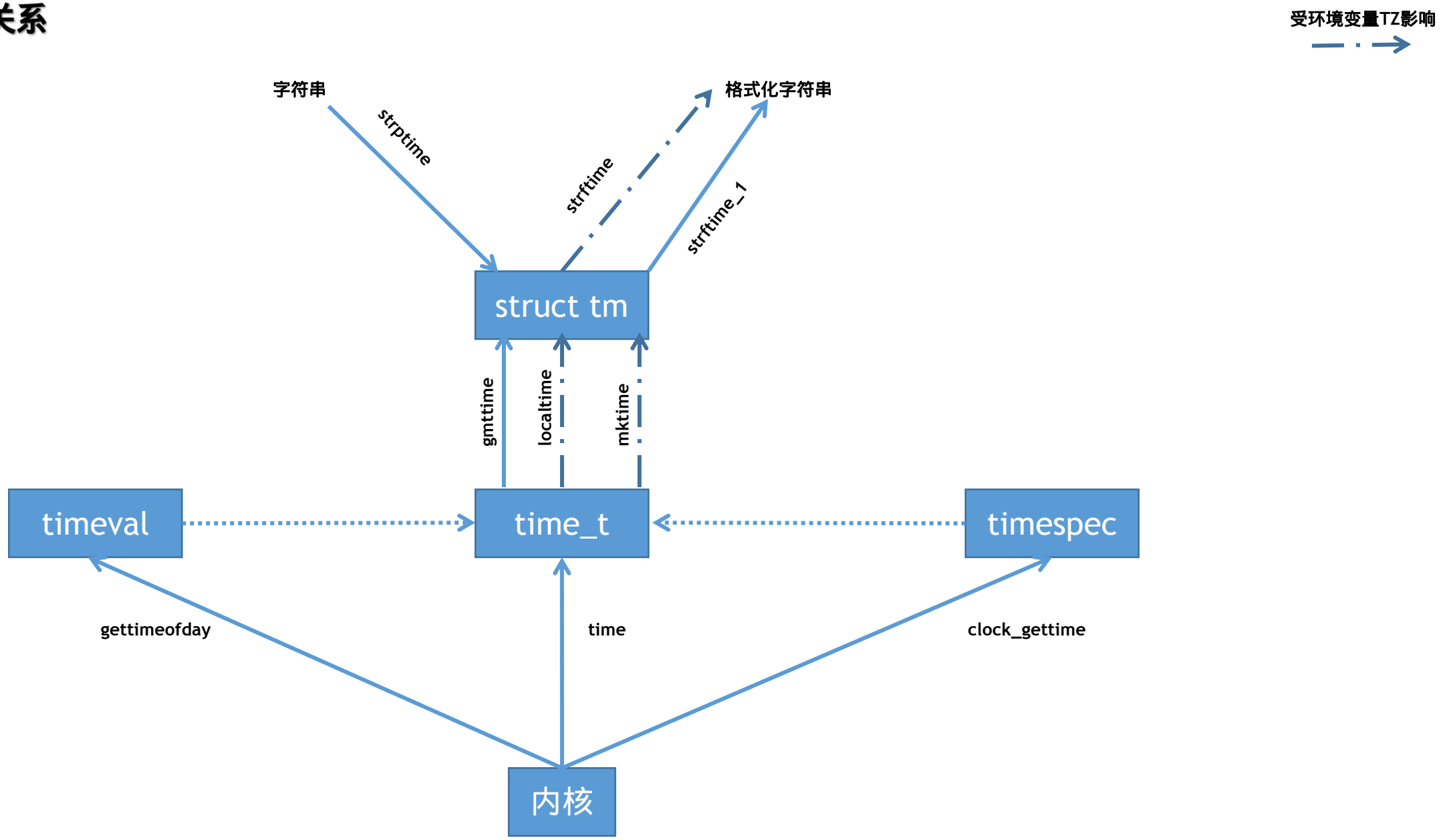
5.获取微秒精度的当前时间
int gettimeofday(struct timeval *tp, void *tzp); //总是返回 0
第二个参数只能设置为NULL

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

```
{
    CLOCK_REALTIME; //实时系统时间
    CLOCK_MONOTONIC; //不带负跳数的实时系统时间
    CLOCK_PROCESS_CPUTIME_ID; //调用进程的CPU时间
    CLOCK_THREAD_CPUTIME_ID; //调用线程的CPU时间
}
```

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

时间函数关系


受环境变量TZ影响
字符串
strptime
strftime
格式化字符串
strftime_1
struct tm
gmttime
localtime
mktime
timeval
time_t
timespec
gettimeofday
time
clock_gettime
内核

时间函数转换

```
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 */
};
```

1.
struct tm *gmtime(const time_t *p)

struct tm *localtime(const time_t *p)

成功，返回指针，失败，返回NULL

localtime转化成本地之间，考虑到时区　、夏令等因素
gmtime转换成标准时间

2. time_t mktime(struct tm *p)
成功，返回日历时间，失败，返回-1

3. size_t strftime(char *buf, size_t maxsize, const char *format, const struct tm *p)
size_t strftime_1(char *buf, size_t maxsize, const char *format, const struct tm *p, locale_t locale)

成功，返回存入数组的字节数，失败，返回0

4.char *strptime(const char *buf, const char *format, struct tm *tmptr)
成功，返回指向上次解析的字符的下一个字符的指针，否则，返回NULL