Chapter 6.
System Data Files & Info

朱金辉

华南理工大学软件学院

1. Introduction

- System Files:
  - password file: /etc/passwd
  - group file: /etc/group
- A portable interface for application programs to access these system files
- Time and date functions

2. Password File

- ASCII text: /etc/passwd
- passwd structure in <pwd.h>

Description	struct passwd member	POSIX.1	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10
user name	char *pw_name	•	•	•	•	•
encrypted password	char *pw_passwd		•	•	•	•
numerical user ID	uid_t pw_uid	•	•	•	•	•
numerical group ID	gid_t pw_gid	•	•	•	•	•
comment field	char *pw_gecos		•	•	•	•
initial working directory	char *pw_dir	•	•	•	•	•
initial shell (user program)	char *pw_shell	•	•	•	•	•
user access class	char *pw_class		•		•	
next time to change password	time_t pw_change		•		•	
account expiration time	time_t pw_expire		•		•	

Functions to fetch entries

- #include <sys/types.h>
- #include <pwd.h>
- struct passwd *getpwuid(uid_t *uid*);

- struct passwd *getpwnam(const char **name*);
- Return: pointer if OK, NULL on error

Go through passwd file

- #include <sys/types.h>
- #include <pwd.h>
- struct passwd *getpwent(void);
- Returns: pointer if OK, NULL on error/EOF
- void setpwent(void);
- void endpwent(void);

Prog 6.2: Implement getpwnam

```
#include <sys/types.h>
#include <pwd.h>
#include <stddef.h>
#include <string.h>
struct passwd * getpwnam(const char *name) {
    struct passwd  *ptr;
    setpwent();
    while ( (ptr = getpwent()) != NULL) {
        if (strcmp(name, ptr->pw_name) == 0)
            break;     /* found a match */
    }
    endpwent();
    return(ptr);/* ptr is NULL if no match found */
}
```

3. Shadow Passwords

- Encrypted passwords → /etc/shadow or /etc/master.passwd
- Shadow is readable only by root
- /etc/passwd is world-readable
- Cannot access encrypted passwd data for guessing the real passwords!

4. Group File

- ASCII text: /etc/group
- group structure in <grp.h>

Description	struct group member	POSIX.1	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10
group name	`char *gr_name`	•	•	•	•	•
encrypted password	`char *gr_passwd`		•	•	•	•
numerical group ID	`int gr_gid`	•	•	•	•	•
array of pointers to individual user names	`char **gr_mem`	•	•	•	•	•

Lookup group entry

```
#include <sys/types.h>

#include <grp.h>

struct group *getgrgid(gid_t gid);

struct group *getgrnam(const char *name);
```

Return: pointer if OK, NULL on error

Search group file

```
#include <sys/types.h>

#include <grp.h>

struct group *getgrent(void);
```

Returns: pointer if OK, NULL on error/EOF

```
void setgrent(void);

void endgrent(void);
```

6. Implementation Differences

Information	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10
account information	`/etc/passwd`	`/etc/passwd`	Directory Services	`/etc/passwd`
encrypted passwords	`/etc/master.passwd`	`/etc/shadow`	Directory Services	`/etc/shadow`
hashed password files?	yes	no	no	no
group information	`/etc/group`	`/etc/group`	Directory Services	`/etc/group`

Figure 6.5 Account implementation differences

7. Other Data Files

Description	Data file	Header	Structure	Additional keyed lookup functions
passwords	/etc/passwd	<pwd.h>	passwd	getpwnam, getpwuid
groups	/etc/group	<grp.h>	group	getgrnam, getgrgid
shadow	/etc/shadow	<shadow.h>	spwd	getspnam
hosts	/etc/hosts	<netdb.h>	hostent	getnameinfo, getaddrinfo
networks	/etc/networks	<netdb.h>	netent	getnetbyname, getnetbyaddr
protocols	/etc/protocols	<netdb.h>	protoent	getprotobyname, getprotobynumber
services	/etc/services	<netdb.h>	servent	getservbyname, getservbyport

Figure 6.6 Similar routines for accessing system data files

8. Login Accounting

- utmp: currently logged-in users
- wtmp: all logins and logouts

```
struct utmp {
   char ut_line[8]; /* tty line */
   char ut_name[8]; /* login name */
   long ut_time; /* secs since Epoch */
};
```

9. System Identification

- #include <sys/utsname.h>
- int uname(struct utsname *name);
- Returns: non-negative value if OK, −1 on error

```
struct utsname {
  char sysname[];/* name of the operating system */
  char nodename[];/* name of this node */
  char release[];/* current release of operating system */
  char version[];/* current version of this release */
  char machine[];/* name of hardware type */
};
```

10. Time and Date Routines

- #seconds since Epoch: 00:00:00 1970/1/1, UTC

```
#include <time.h>
time_t time(time_t *calptr);
```

- Returns: value of time if OK, -1 on error
- time is stored in calptr if not NULL

Time functions

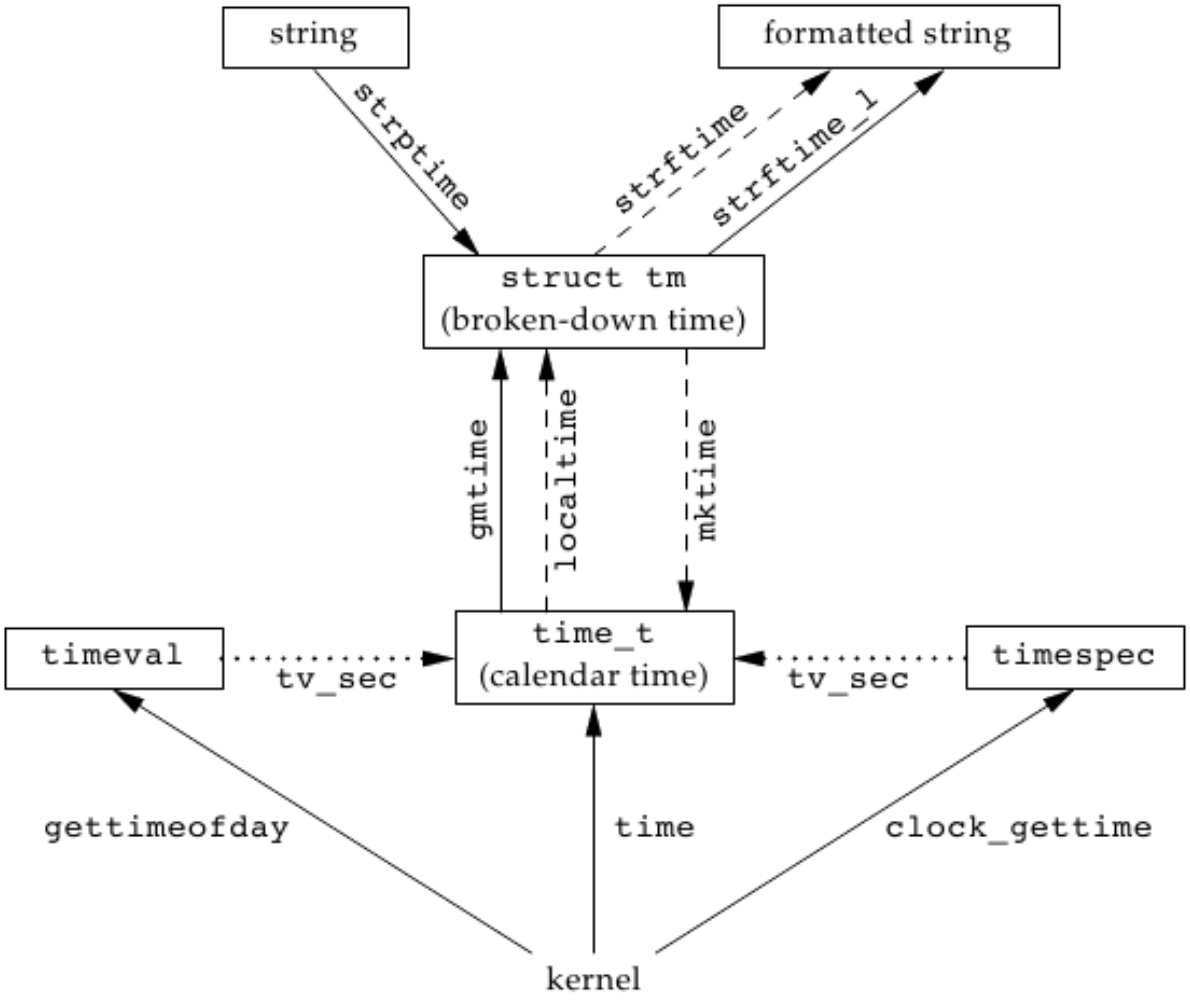

Figure 6.9 Relationship of the various time functions

Broken down time

struct tm {

- int tm_sec; /* secs after the minute: [0, 60] */
- int tm_min; /* minutes after the hour: [0, 59] */
- int tm_hour; /* hours after midnight: [0, 23] */
- int tm_mday; /* day of month: [1, 31] */
- int tm_mon; /* month of year: [0, 11] */
- int tm_year; /* years since 1900 */
- int tm_wday; /* days since Sunday: [0, 6] */
- int tm_yday; /* days since Jan 1: [0, 365] */
- int tm_isdst; /* daylight saving time flag: <0, 0, >0 */

};

Time functions

```
#include <time.h>

struct tm *gmtime(const time_t *calptr);

struct tm *localtime(const time_t *calptr);
```

- Return: pointer to broken-down time

```
time_t mktime(struct tm *tmptr);
```

- Returns: calendar time if OK, -1 on error

Time function

- size_t strftime(char **buf*, size_t *maxsize*, const char **format*, const struct tm **tmptr*);
- Returns: #char stored in buf if room, else 0
- Time value from *tmptr* is formatted according to *format* and stored in *buf* of size *maxsize*, if there is enough room, otherwise 0 is returned.

Time formats

Format	Description	Example
%a	abbreviated weekday name	Thu
%A	full weekday name	Thursday
%b	abbreviated month name	Jan
%B	full month name	January
%c	date and time	Thu Jan 19 21:24:52 2012
%C	year/100: [00−99]	20
%d	day of the month: [01−31]	19
%D	date [MM/DD/YY]	01/19/12
%e	day of month (single digit preceded by space) [1−31]	19
%F	ISO 8601 date format [YYYY–MM–DD]	2012-01-19
%g	last two digits of ISO 8601 week-based year [00−99]	12
%G	ISO 8601 week-based year	2012
%h	same as %b	Jan
%H	hour of the day (24-hour format): [00−23]	21
%I	hour of the day (12-hour format): [01−12]	09

Figure 6.10 Conversion specifiers for strftime

Using the strftime function Program 6.11

```
time_t t;
struct tm *tmp;
char buf1[16];
char buf2[64];
time(&t);
tmp = localtime(&t);
if (strftime(buf1, 16, "time and date: %r, %a %b %d, %Y", tmp) == 0)
    printf("buffer length 16 is too small\n");
else
    printf("%s\n", buf1);
if (strftime(buf2, 64, "time and date: %r, %a %b %d, %Y", tmp) == 0)
    printf("buffer length 64 is too small\n");
else
    printf("%s\n", buf2);
```

Program 6.11 result

```
$ ./a.out
buffer length 16 is too small
time and date: 11:12:35 PM, Thu Jan 19, 2012
```