Chapter 6.
System Data Files and Information

System Programming

http://www.cs.ccu.edu.tw/~pahsiung/courses/pd

熊博安

國立中正大學資訊工程學系

pahsiung@cs.ccu.edu.tw

(05)2720411 ext. 33119

Class: EA-104

Office: EA-512

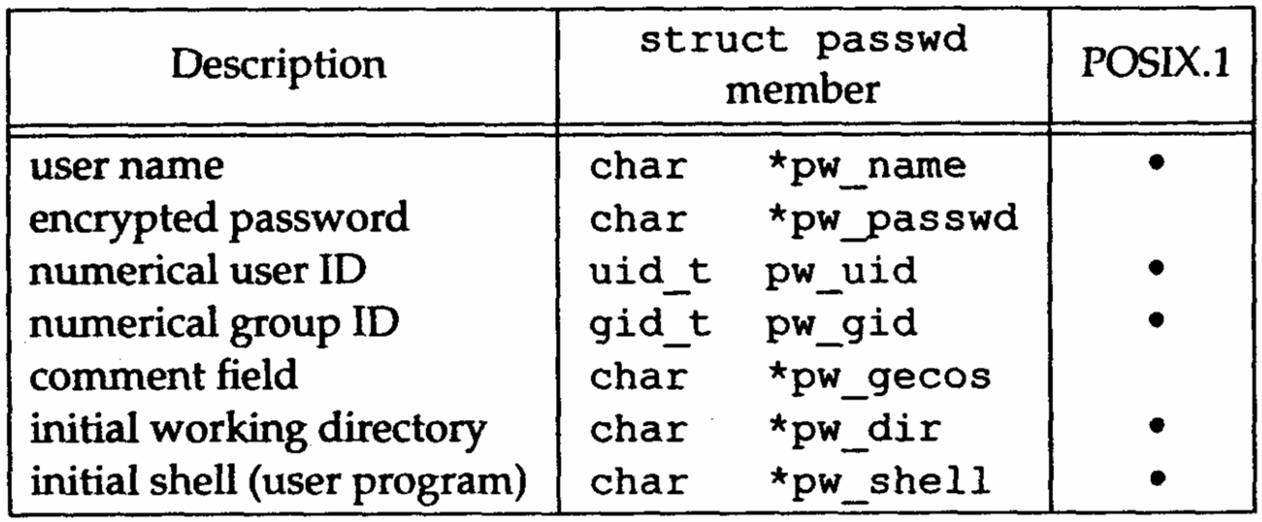
Textbook: Advanced Programming in the UNIX Environment, 2nd Edition, W. Richard Stevens and Stephen A. Rago

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

Password File

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



Description	struct passwd member	POSIX.1
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	•

Figure 6.1 Fields in /etc/passwd file.

Password File

- 3 more fields in FreeBSD 5.2.1 and Mac OS X 10.3
  - user access class: char *pw_class
  - next time to change password: time_t pw_change
  - account expiration time: time_t pw_expire

Password File

- root: superuser, UID=0
  - root:x:0:0:root:/root:/bin/bash
- device account, no login allowed
  - squid:x:23:23::/var/spool/squid:/dev/null
- guest, no priviledges
  - nobody:x:65534:65534:Nobody:/home/bin/sh
- normal user account
  - sar:x:205:105:Stephen Rago:/home/sar:/bin/bash

Password File

- Comment field:
  - Steve Rago, SF 5-121, 555-1111, 555-2222
  - Name, Office, Phone Numbers, etc.
- vipw command
  - edits password file
  - serializes changes to password file

Functions to fetch entries

- #include <pwd.h>

used by ls

- struct passwd *getpwuid(uid_t *uid*);

used by login

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

Go through passwd file

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

Fig. 6.2: Implement getpwnam

```c
#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 */
}
```

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!

Shadow Passwords

long int
in Linux

Description	struct spwd member
user login name	char *sp_namp
encrypted password	char *sp_pwdp
days since Epoch of last password change	int sp_lstchg
days until change allowed	int sp_min
days before change required	int sp_max
days warning for expiration	int sp_warn
days before account inactive	int sp_inact
days since Epoch when account expires	int sp_expire

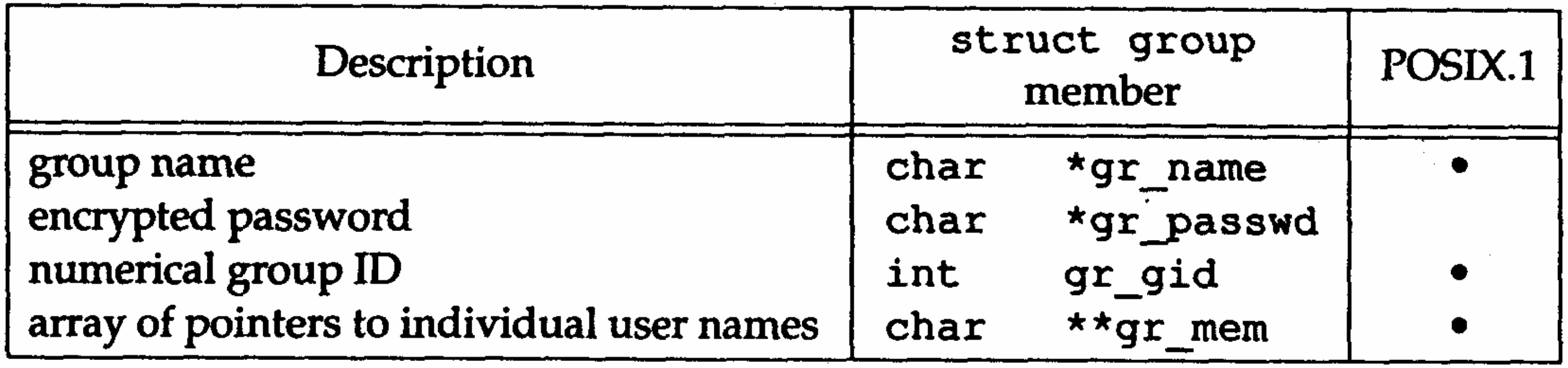
Shadow Passwords

- Linux 2.4.22 and Solaris 9
  - Separate set of functions to access shadow passwords

```
#include <shadow.h>
struct spwd *getspnam(const char *name);
struct spwd *getspent(void);
    Return: pointer if OK, NULL on error
void setspent(void);
void endspent(void);
```

Group File

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



Description	struct group member		POSIX.1
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	•

Figure 6.2 Fields in /etc/group file.

Lookup group entry

```
#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 <grp.h>

struct group *getgrent(void); /* read next entry */

  Returns: pointer if OK, NULL on
  error/EOF

void setgrent(void); /* rewind group file */

void endgrent(void); /* close group file */
```

Supplementary Group IDs

- POSIX.1 requirement
- A user could belong to a group with GID in password file, and
- A user could at the same time belong to up to 16 additional supplementary groups
- No need to change groups using newgrp() as we did before

Supplementary Group IDs

```
#include <unistd.h>
int getgroups(int gidsetsize, gid_t grouplist[]);
```

Returns: #supp GIDs if OK, -1 on error

```
#include <grp.h> /* on Linux */
#include <unistd.h> /*on FreeBSD, Mac OS X, Solaris*/
int setgroups(int ngroups, const gid_t grouplist[]);

#include <grp.h> /* on Linux, Solaris */
#include <unistd.h> /*on FreeBSD, Mac OS X */
int initgroups(const char *username, gid_t basegid);
```

Both return: 0 if OK, -1 on error

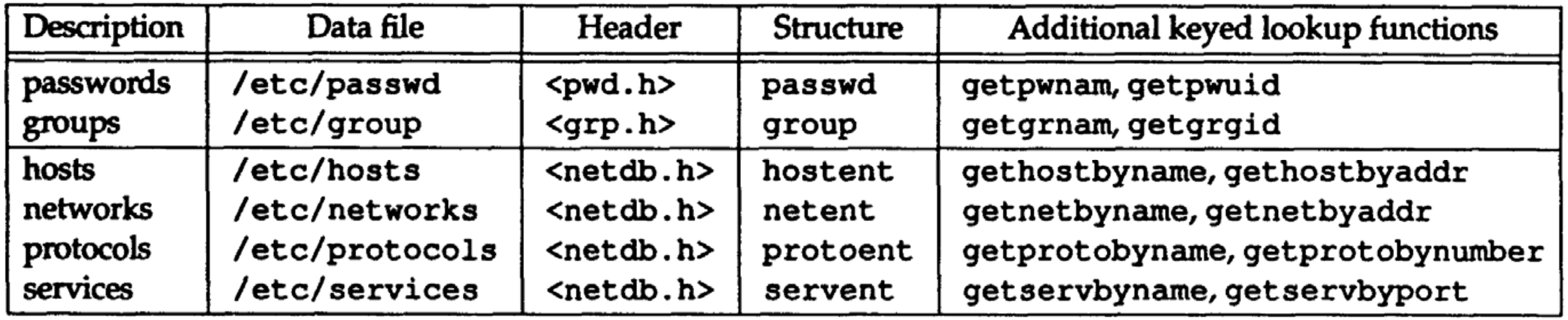
Supplementary Group IDs

- getgroups
  - Fills **`grouplist`** with **`gidsetsize`** group IDs
- setgroups
  - set supplementary GID for calling process
- initgroups
  - reads entire group file (**`getgrent`**, **`setgrent`**, **`endgrent`**)
  - calls **`setgroups`** to initialize supplementary GIDs
  - **`basegid`** (from password file) is also included
  - called by login

Implementation Differences

Info	FreeBSD 5.2.1	Linux 2.4.22	Mac OS X 10.3	Solaris 9
Account Info	/etc/passwd	/etc/passwd	netinfo	/etc/passwd
Encrypted Password	/etc/master.passwd	/etc/shadow	netinfo	/etc/shadow
Hashed Password Files?	Yes /etc/pwd.db /etc/spwd.db	No	No	No
Group Info	/etc/group	/etc/group	netinfo	/etc/group

Other Data Files (Fig. 6.6)



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
hosts	/etc/hosts	<netdb.h>	hostent	gethostbyname, gethostbyaddr
networks	/etc/networks	<netdb.h>	netent	getnetbyname, getnetbyaddr
protocols	/etc/protocols	<netdb.h>	protoent	getprotobyname, getprotobynumber
services	/etc/services	<netdb.h>	servent	getservbyname, getservbyport
shadow	/etc/shadow	<shadow.h>	spwd	getspnam

Similar routines to access the data files: **get**, **set**, **end**

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 */
};
```

System Identification

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

struct utsname {
  char sysname[];    /* name of OS */
  char nodename[];  /* name of node */
  char release[];     /* OS release */
  char version[];     /* OS version */
  char machine[];    /* name of hardware type */
};
```

System Identification

```
#include <unistd.h>
int gethostname(char *name, int namelen);
    Returns: 0 if OK, -1 on error
```

- Null-terminated name if enough space
- POSIX.1: HOST_NAME_MAX

Time and Date Routines

- #seconds since Epoch: 00:00:00 1970/1/1, UTC
- Different from other OS:
  - keeping time in UTC, instead of local time
  - automatic time conversions
    - E.g.: daylight saving time
  - keeping time and date as a single quantity

time & gettimeofday functions

```
#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

```
struct timeval {
  time_t tv_sec; /* sec */
  long tv_usec; /*microsec*/
};
```

```
#include <sys/time.h>
int gettimeofday(struct timeval *restrict tp,
                 void *restrict tzp);
```

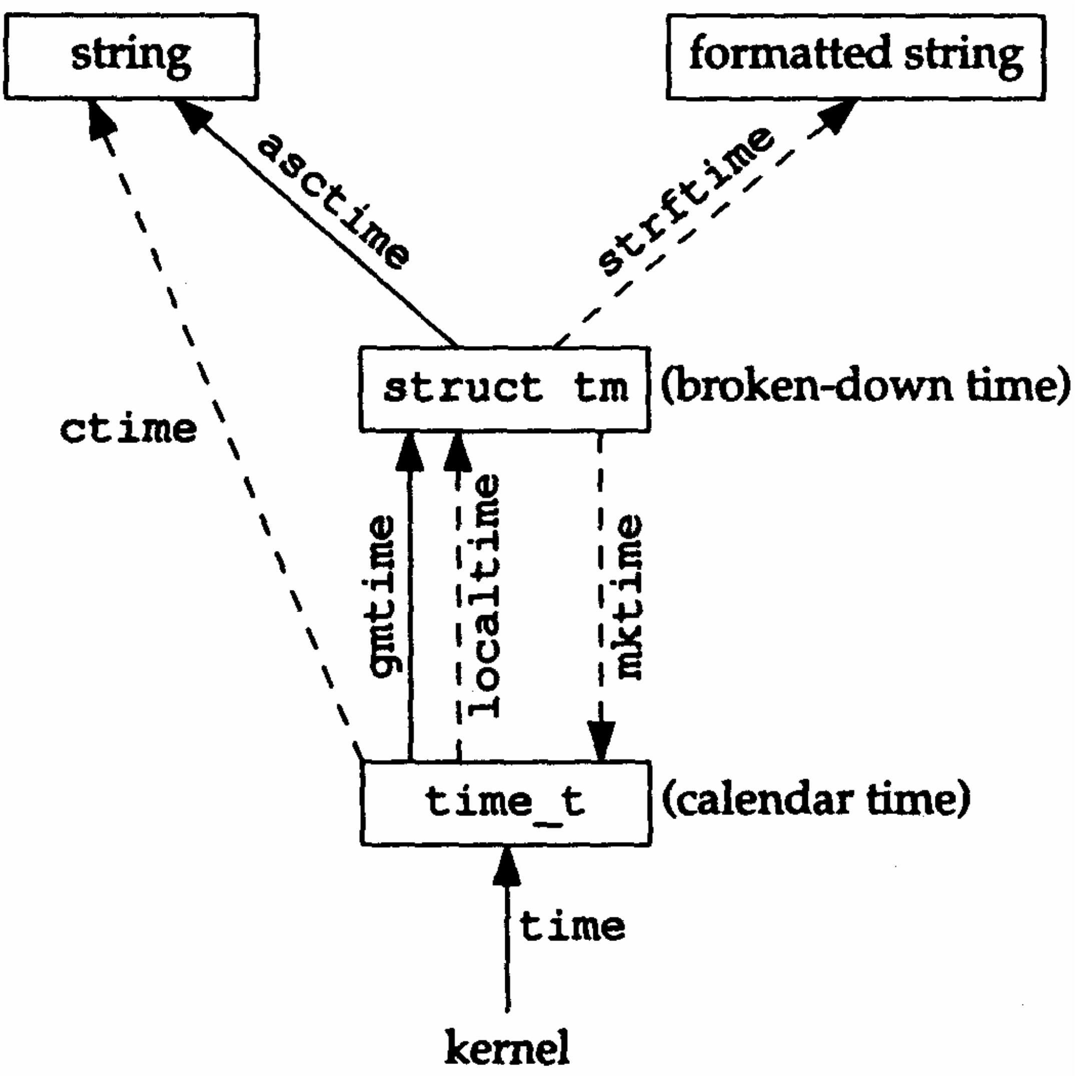
NULL

Returns: 0 always

Time functions

Affected by TZ env var:

- localtime,
- mktime,
- ctime,
- strftime



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 1990 */
- 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

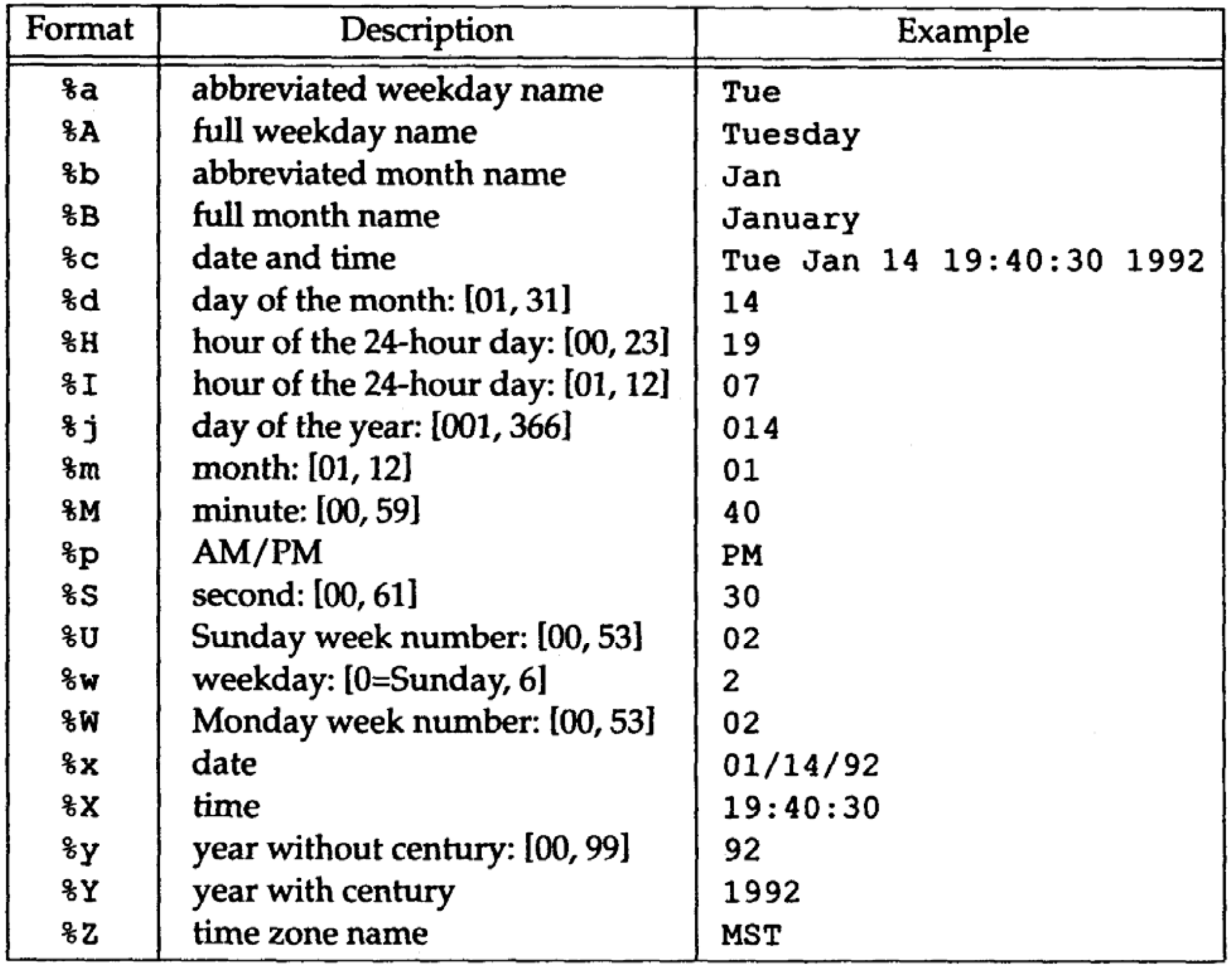
Time functions

- date: Tue Jan 14 17:49:03 1992\n\0
- totally 26-bytes
- char *asctime(const struct tm **tmptr*);
- char *ctime(const time_t **calptr*);
- Return: pointer to string

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.

Conversion Specifiers for Time Formats in strftime()



Format	Description	Example
`%a`	abbreviated weekday name	`Tue`
`%A`	full weekday name	`Tuesday`
`%b`	abbreviated month name	`Jan`
`%B`	full month name	`January`
`%c`	date and time	`Tue Jan 14 19:40:30 1992`
`%d`	day of the month: [01, 31]	`14`
`%H`	hour of the 24-hour day: [00, 23]	`19`
`%I`	hour of the 24-hour day: [01, 12]	`07`
`%j`	day of the year: [001, 366]	`014`
`%m`	month: [01, 12]	`01`
`%M`	minute: [00, 59]	`40`
`%p`	AM/PM	`PM`
`%S`	second: [00, 61]	`30`
`%U`	Sunday week number: [00, 53]	`02`
`%w`	weekday: [0=Sunday, 6]	`2`
`%W`	Monday week number: [00, 53]	`02`
`%x`	date	`01/14/92`
`%X`	time	`19:40:30`
`%y`	year without century: [00, 99]	`92`
`%Y`	year with century	`1992`
`%Z`	time zone name	`MST`

Time Zone (TZ env var)

- Functions affected by TZ:
    - localtime(), mktime(), ctime(), strftime()
- E.g.: TZ=EST5EDT