Contents

System Data Files and Info

- Objective
  - System Data Files
    - E.g., /etc/passwd and /etc/group
  - Portable Interfaces to System Data Files
    - Formats other than ASCII text files
- /etc/passwd – local machine or NIS DB
  - User database in POSIX.1 <pwd.h> (Fig.6.1)
  - root:jhexh38fhajck:0:1:Super-User:/root:/bin/tcsh
  - Login-name, encrypted passwd, numeric user-ID, numeric group ID, comment, home dir, shell program

System Data Files and Info

- /etc/passwd (continue)
  - Superuser *root* – UID = 0
  - One-way encryption algorithm for passwd → 13 printable chars from 64-char sets [a-zA-Z0-9./].
    - nobody:*:65534:65534:SunOS 4.x Nobody:/
    - /bin/sh for default
  - Command finger

System Data Files and Info

```
#include <sys/types.h>
#include <pwd.h>
struct passwd *getpwuid(uid_t uid);
struct passwd *getpwnam(const char
  *name);
```

- getpwuid() used by Command *l*s
- getpwnam() used by the login program
  - Both use a static variable for returning
- POSIX.1

System Data Files and Info

```
#include <sys/types.h>
#include <pwd.h>
struct passwd *getpwent(void);
void setpwent(void);
void endpwent(void);
```

- No order in the returned pwd entries.
- setpwent()/endpwent rewind/close these files.
- Non-POSIX.1

- Program 6.1 – Page 148
  - getpwnam

System Data Files and Info

- /etc/shadow – shadow passwd file
  - /etc/passwd
    - root:x:0:1:Super-User:/root:/bin/tcsh
    - with “x” indicated for passwd
  - Contents
    - Username, passwd, passwd aging
    - SVR4: /etc/shadow, 4.3+BSD: /etc/master.passwd
    - Readable by set-usr-ID login/passwd programs
  - Rationale: avoid a brute force approach in trying passwds

System Data Files and Info

- /etc/group – the group database
  - nuucp::9:root,nuucp
  - Figure 6.2 – Page 149
    - gr_passwd not in POSIX.1 (in SVR4 & 4.3+BSD)

```
#include <sys/types.h>
#include <grp.h>
struct group *getgrgid(gid_t gid);
struct group *getgrnam(const char *name);
```

  - A static variable for returned values.

System Data Files and Info

```
#include <sys/types.h>
#include <grp.h>
struct group *getgrent(void);
void setgrent(void);
void endgrent(void);
```

- setgrent() open (if not) and rewind the group file.
- endgrent() close the group file.

System Data Files and Info – Supplementary Group IDs

- Introduction of supplementary group ID's – 4.2BSD
  - newgrp is the way to change gid since Version 7
  - They all can be used to check for file access permissions
  - Optional in POSIX.1, NGROUP_MAX (16 in common)

System Data Files and Info – Supplementary Group IDs

```
#include <sys/types.h>
#include <unistd.h>
int getgroups(int gidsetsize, gid_t grouplist[]);
int setgroups(int ngroups, const gid_t
    grouplist[]);
int initgroups(const char *usrname, gid_t
    basegid);
```

- gidsetsize = 0 → only number is returned.
- Only superusers can call setgroups() and initgroups() – SVR4&4.3BSD
  - Called by the login program

System Data Files and Info

- BSD Networking Software
  - /etc/services – network services
  - /etc/protocols – protocols
  - /etc/networks – networks
- General Principle to the Interfaces
  - A get function to read the next record
  - A set function to rewind the file
  - An end function to close the file
  - Keyed lookup functions if needed.
  - Figure 6.3 – Page 153
    - Routines for System File Access

Login Accounting

- /etc/utmp → /var/adm/utmp in SVR4 (or /var/run/utmp in 4.3+BSD)
  - ut_line, ut_name, ut_time (in sec since Epoch)
  - Updated by the login program, erased by init
- /etc/wtmp → /var/adm/wtmp in SVR4 (or /var/log/wtmp in 4.3+BSD)
  - Updated by the login and init programs, reboot
- Related Commands: last, who, etc

```
> last | grep ktw
ktw      pts/26       austin.csie.ntu. Fri Apr 12 18:22   still logged in
ktw      pts/5        pc210.ice.ntnu.e Thu Apr 11 10:25 - 10:36  (00:10)
```

- /etc/?tmp entries: 20 bytes (Ver 7) → 350 bytes (SVR4)

System Identification

```
#include <sys/utsname.h>
int uname(struct utsname *name);
    struct utsname {
        char sysname[9] /* name of OS */
        char nodename[9]; /* name of the node */
        char release[9]; /* current release of the OS */
        char version[9]; /* current ver of the release */
        char machine[9]; /* name of the HW type */     }
```

- sysconf()

```
#include <sys/utsname.h>
int gethostname(char *name, int namelen);
```

- Domain name of the host on a TCP/IP network – BSD systems
- MAXHOSTNAMELEN in <sys/param.h>

System Identification

- Commands: hostname, sethostname (/etc/rc), uname
- $ uname –a

  SunOS ntucsa 5.6 Generic_105181-26 sun4u sparc sun4u
  - Nodename is not good to reference on a network.
  - Most versions of System V has this info compiled into the kernel when the kernel is built.

Time and Date Routines

- Time Values
  - Calendar time
    - In seconds since the Epoch (00:00:00 January 1, 1970, Coordinated Universal Time, i.e., UTC)
    - type *time_t*
  - Remark: Times in Unix
    - Keeping time in UTC
    - Automatic handling of conversions, such as daylight saving time
    - Keeping of time and date as a single quantity.

Time and Date Routines

- Time Values (continued)
  - Process time
    - In clock ticks (divided by CLK_TCK -> secs)
    - type *clock_t*
    - Clock time, user/system CPU time

    > time grep _POSIX_SOURCE */*.h > /dev/null

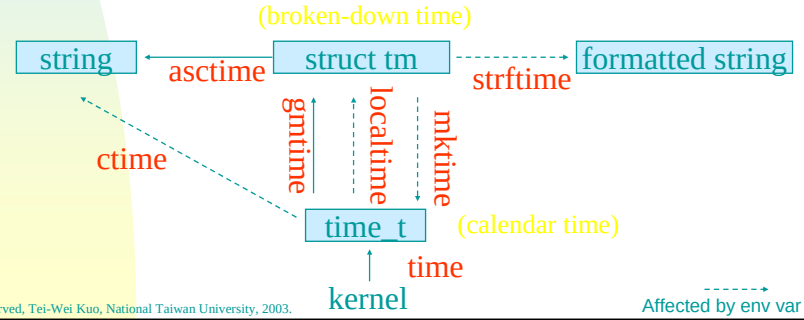
    0.25u 0.25s 0:03.51 14.2%

Time and Date Routines

#include <time.h>

time_t time(time_t *calptr);

- As a func in BSD, call gettimeofday (1us)
- Time initialization: settimeofday (BSD, 1us), stime (SVR4)



Time and Date Routines

#include <time.h>

struct tm *gmtime(const time_t *calptr);

struct tm *localtime(const time_t *calptr);

```
struct tm { /* borken-down time */
        int tm_sec;   /* [0, 61], >= 59 for leap seconds*/
        int tm_min;   /* [0, 59] */
        int tm_hour;  /* [0, 23] */
        int tm_mday; /* [1, 31] */
        int tm_mon;   /* [0, 11] */
        int tm_year;   /* years since 1900 */
        int tm_wday;  /* days since Sunday: [0, 6] */
        int tm_yday;   /* days since January 1: [0, 365] */
        int tm_isdst;   /* daylight saving time flag: > 0, 0, < 0 (not available) */
};
```

- localtime() → local time, gmtime() → UTC time

Time and Date Routines

#include <time.h>

time_t mktime(struct tm *tmptr);

char *asctime(const struct tm *tmptr);

char *ctime(const time_t *calptr);

size_t strftime(char *buf, size_t maxsize, const char *format, const struct tm *tmptr);

- strftime() returns 0 if the size of the buf does not fit!
- Figure 6.5 – Page 158 (Conversion Specifiers)
  - Tue Jan 14 19:40:30 MST 1992

- Remark: env var TZ= → UTC is normally used for ------→