

Ateneo de Manila University

Getting System Information



Department of Information Systems and
Computer Science

S.Y. 2001-2002

<http://sysads.ateneo.net/wyu/>

wyy@admu.edu.ph

System Information

- ★ UNIX systems provide system calls for accessing certain user and system information such as:
 - User and Group
 - Usage Accounting
 - System Identification
 - Date and Time Services
- ★ enables applications developers to utilize the UNIX system is performing certain tasks
- ★ provides system level data structures for manipulating this data

Section I

Password and Group Files

Password

- ★ Unix system provides a mechanism for accessing system related information
- ★ defined in the `pwd.h` header file
- ★ the `passwd` structure is used to return user information:

```
struct passwd {
    char    *pw_name;      /* user name */
    char    *pw_passwd;   /* user password */
    uid_t   pw_uid;       /* user id */
    gid_t   pw_gid;       /* group id */
    char    *pw_gecos;    /* real name */
    char    *pw_dir;      /* home directory */
    char    *pw_shell;    /* shell program */
};
```

```
★ struct passwd *getpwnam(const char * name);
```

- returns the `passwd` structure based on the username
- returns NULL if an error occurs

```
★ struct passwd *getpwuid(uid_t uid);
```

- returns the `passwd` structure based on the user ID
- returns NULL if an error occurs

```
★ void setpwent(void);
```

```
★ struct passwd *getpwent(void);
```

```
★ void endpwent(void);
```

- goes through all the entries in the `/etc/passwd` file
- similar behavior to `readdir`
- returns NULL if an error occurs

Group

★ defined in the `grp.h` header file

★ the `group` structure is used to return user information:

```
struct group {  
    char    *gr_name;    /* group name */  
    char    *gr_passwd; /* group password */  
    gid_t   gr_gid;     /* group id */  
    char    **gr_mem;   /* group members */  
};
```

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

★ `struct group *getgruid(gid_t gid);`

★ `void setgrent(void);`

★ `struct group *getgrent(void);`

```
★ void endgrent(void);
```

- similar behavior to the **passwd** functions

```
★ int getgroups(size_t size, gid_t *list);
```

```
★ int setgroups(size_t size, const gid_t *list);
```

- set and get the current user's supplementary groups and returns their GIDs to an array defined by `list`
- on error a -1 is returned by both functions

Current User's Data

★ `gid_t getgid(void);`

★ `gid_t getegid(void);`

★ `int setregid(gid_t rgid, gid_t egid);`

★ `int setegid(gid_t egid); int setgid(gid_t gid);`

– gets/sets the real and effective group ID of the current user

– real GID corresponds to the GID of the corresponding process while the effective GID corresponds to the set ID bit on the file being executed

★ `pid_t getuid(void);`

★ `pid_t geteuid(void);`

– gets the real and effective user ID of the current user

– real UID corresponds to the UID of the corresponding process while the effective UID corresponds to the set ID bit on the file being executed

Section II

System Information

Login Accounting

★ defined in the `utmp.h` header file

★ the `utmp` structure is used to return user information:

```
struct utmp {
    short ut_type;           /* type of login */
    pid_t ut_pid;           /* pid of login process */
    char ut_line[UT_LINESIZE]; /* device name of tty - "/dev/" */
    char ut_id[4];          /* init id or abbrev. ttyname */
    char ut_user[UT_NAMESIZE]; /* user name */
    char ut_host[UT_HOSTSIZE]; /* hostname for remote login */
    struct exit_status ut_exit /* The exit status of a process
                               marked as DEAD_PROCESS. */

    long ut_session;        /* session ID, used for windowing*/
    struct timeval ut_tv;    /* time entry was made. */
    int32_t ut_addr_v6[4];  /* IP address of remote host. */
    char pad[20];           /* Reserved for future use. */
};

struct exit_status {
    short int e_termination; /* process termination status. */
    short int e_exit;        /* process exit status. */
};
```

★ `void setutent(void);`

★ `struct utmp *getutent(void);`

★ `void endutent(void);`

- similar behavior to the **passwd** functions

★ `struct utmp *getutid(struct utmp *ut);`

- searches forward from the current file position in the utmp file based upon `ut`
- if `ut->ut_type` is `RUN_LVL`, `BOOT_TIME`, `NEW_TIME`, or `OLD_TIME`, `getutid()` will find the first entry whose `ut_type` field matches `ut->ut_type`
- if `ut->ut_type` is one of `INIT_PROCESS`, `LOGIN_PROCESS`, `USER_PROCESS`, or `DEAD_PROCESS`, `getutid()` will find the first entry whose `ut_id` field matches `ut->ut_id`

★ `struct utmp *getutline(struct utmp *ut);`

- searches forward from the current file position in the utmp file
- scans entries whose `ut_type` is `USER_PROCESS` or `LOGIN_PROCESS` and returns the first one whose `ut_line` fields match

System Identification

- ★ defined in the `sys/utsname.h` header file
- ★ the `utsname` structure is used to return user information:

```
struct utsname {
    char sysname[SYS_NMLN];
    char nodename[SYS_NMLN];
    char release[SYS_NMLN];
    char version[SYS_NMLN];
    char machine[SYS_NMLN];
#ifdef _GNU_SOURCE
    char domainname[SYS_NMLN];
#endif
};
```

```
★ int uname(struct utsname *buf);
```

- returns system information in the structure pointed to by buf
- returns -1 if an error occurs

```
★ int gethostname(char *name, size_t len);
```

```
★ int sethostname(const char *name, size_t len);
```

- sets/gets the system hostname with name
- returns -1 if an error occurs

Section III

Date and Time

Date and Time Routines

★ defined in the `time.h` header file

★ `time_t time(time_t *t);`

- returns the time in seconds since the Epoch
- if `t` is a non-NULL value the time is stored in `t`
- on error `((time_t)-1)` is returned

★ `struct tm *gmtime(const time_t *timep);`

★ `struct tm *localtime(const time_t *timep);`

★ `time_t mktime(struct tm *timeptr);`

- converts time to the `tm` structure

```
struct tm
{
    int tm_sec;    /* seconds */
```

```
    int tm_min;    /* minutes */
    int tm_hour;  /* hours */
    int tm_mday;  /* day of the month */
    int tm_mon;   /* month */
    int tm_year;  /* year */
    int tm_wday;  /* day of the week */
    int tm_yday;  /* day in the year */
    int tm_isdst; /* DST */
};
```

- `gmtime` converts the resulting time in UTC
- `localtime` converts the resulting time in local time
- `mktime` does the reverse of the previous two functions

```
* char *asctime(const struct tm *timeptr);
```

- converts `tm` time to a string in the format of the **date** command

- returns NULL if an error occurred

```
* char *ctime(const time_t *timep);
```

- converts `time_t` time to a string in the format of the **date** command
- returns NULL if an error occurred



Copyright © 2000-2001 by William Emmanuel S. Yu. This material may be distributed only subject to the terms and conditions set forth in the Open Content License, v1.0 or later (the latest version is presently available at <http://opencontent.org/opl.shtml>).