4 시스템 정보

학습목표

- 유닉스 시스템 정보를 검색하는 함수를 사용할 수 있다.
- 사용자 관련 정보를 함수를 이용해 검색할 수 있다.
- 시스템의 시간을 관리하는 함수를 사용할 수 있다.

목차

유닉스 시스템 관련정보

- 시스템에 설치된 운영체제에 관한 정보
- 호스트명 정보
- 하드웨어 종류에 관한 정보
- 하드웨어에 따라 사용할 수 있는 자원의 최댓값
  - 최대 프로세스 개수
  - 프로세스당 열 수 있는 최대 파일 개수
  - 메모리 페이지 크기 등

운영체제 기본 정보 검색

시스템에 설치된 운영체제에 대한 기본 정보 검색

```
# uname -a
SunOS hanbit 5.10 Generic_118855-33 i86pc i386 i86pc
```

운영체제명 | 호스트명 | 릴리즈 레벨 | 버전 번호 | 하드웨어 형식명 | CPU명 | 플랫폼명

- 시스템은 인텔PC고 솔라리스 10운영체제가 설치되어 있고, 호스트명은 hanbit

운영체제 정보 검색 함수 : uname(2)

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

int uname(struct utsname *name);
```

- utsname 구조체에 운영체제 정보 저장
  - sysname : 현재 운영체제 이름
  - nodename : 호스트명
  - release : 운영체제의 릴리즈 번호
  - version : 운영체제 버전 번호
  - machine : 하드웨어 아키텍처 이름

```
struct utsname {
  char sysname[_SYS_NMLN];
  char nodename[_SYS_NMLN];
  char release[_SYS_NMLN];
  char version[_SYS_NMLN];
  char machine[_SYS_NMLN];
};
```

[예제 4-1] uname 함수 사용하기

ex4_1.c

```
01  #include <sys/utsname.h>
02  #include <stdlib.h>
03  #include <stdio.h>
04
05  int main(void) {
06      struct utsname uts;
07
08      if (uname(&uts) == -1) {
09          perror("uname");
10          exit(1);
11      }
12
13      printf("OSname : %s\n", uts.sysname);
14      printf("Nodename : %s\n", uts.nodename);
15      printf("Release  : %s\n", uts.release);
16      printf("Version  : %s\n", uts.version);
17      printf("Machine  : %s\n", uts.machine);
18
19      return 0;
20  }
```

```
# ex4_1.out
OSname : SunOS
Nodename : hanbit
Release  : 5.10
Version  : Generic_118855-33
Machine  : i86pc
```

시스템 정보 검색과 설정[1]

시스템 정보 검색과 설정: sysinfo(2) 우분투에서 제공 ×

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

long sysinfo(int command, char *buf, long count);
```

- command에 검색하거나 설정할 명령 지정
- command에 사용할 상수의 범주

상수 범위	예약된 용도
1 ~ 256	• 유닉스 표준에서 정의한 상수 • 정보를 검색(get)하는 데 사용하기 위해 예약된 번호
257 ~ 512	• 유닉스 표준에서 정의한 상수 • 정보를 정의(set)하는 데 사용하기 위해 예약된 번호 • 검색 번호 + 256으로 번호 할당
513 ~ 768	• 솔라리스에서 추가로 정의한 상수 • 정보를 검색(get)하는 데 사용하기 위해 예약된 번호
769 ~ 1024	• 솔라리스에서 추가로 정의한 상수 • 정보를 정의(set)하는 데 사용하기 위해 예약된 번호 • 검색 번호 + 256으로 번호 할당

시스템 정보 검색과 설정[2]

- 유닉스 표준에서 정의한 정보 검색용 상수

상수	설명
SI_SYSNAME(1)	운영체제명을 리턴한다. uname 함수의 sysname 항목과 같은 값이다.
SI_HOSTNAME(2)	uname 함수의 nodename 항목과 같은 값으로, 현재 시스템의 호스트명을 리턴한다.
SI_VERSION(4)	uname 함수의 version 항목과 같은 값을 리턴한다.
SI_MACHINE(5)	하드웨어 형식 값을 리턴한다. uname 함수의 machine 항목과 같은 값이다.
SI_ARCHITECTURE(6)	하드웨어의 명령어 집합 아키텍처(ISA, Instruction Set Architecture) 정보를 리턴한다. 예를 들면, sparc, mc68030, i386 등이다.
SI_HW_SERIAL(7)	하드웨어 장비의 일련번호를 리턴한다. 기본적으로 이 일련번호는 중복되지 않는다. SI_HW_PROVIDER 값과 함께 사용하면 모든 SVR4 업체의 제품을 구별하는 유일한 번호가 된다.
SI_HW_PROVIDER(8)	하드웨어 제조사 정보를 리턴한다.
SI_SRPC_DOMAIN(9)	Secure RPC(Remote Procedure Call) 도메인명을 리턴한다.

시스템 정보 검색과 설정[2]

- 유닉스 표준에서 정의한 정보 설정용 상수

상수	설명
SI_SET_HOSTNAME(258)	호스트명을 설정한다. 이 명령은 root 사용자만 사용할 수 있다.
SI_SET_SPRC_DOMAIN(265)	Secure RPC 도메인을 설정한다.

- 솔라리스에서 정의한 정보 검색용 상수

상수	설명
SI_PLATFORM(513)	하드웨어 플랫폼 모델명을 리턴한다. 예를 들면, 'SUNW,Sun_4_75', 'SUNW,SPARCsystem-600', 'i86pc' 등이다.
SI_ISALIST(514)	현재 시스템에서 실행 가능한 하드웨어 명령어 집합 아키텍처 정보의 목록을 리턴한다.
SI_DHCP_CACHE(515)	부팅 시 DHCP 서버에서 DHCPACK 응답으로 받은 값을 리턴한다.
SI_ARCHITECTURE_32(516), SI_ARCHITECTURE_64(517), SI_ARCHITECTURE_K(518), SI_ARCHITECTURE_NATIVE(519)	32비트 또는 64비트 하드웨어의 명령어 집합 아키텍처 정보를 리턴한다.

[예제 4-2] sysinfo 함수 사용하기(검색)

ex4_2.c

```
01  #include <sys/systeminfo.h>
02  #include <stdlib.h>
03  #include <stdio.h>
04
05  int main(void) {
06      char buf[257];
07
08      if (sysinfo(SI_HW_SERIAL, buf, 257) == -1) {
09          perror("sysinfo");
10          exit(1);
11      }
12      printf("HW Serial : %s\n", buf);
13
14      if (sysinfo(SI_ISALIST, buf, 257) == -1) {
15          perror("sysinfo");
16          exit(1);
17      }
18      printf("ISA List : %s\n", buf);
19
20      return 0;
21  }
```

하드웨어 일련번호 검색

사용가능한 아키텍처 목록검색

```
# ex4_2.out
HW Serial : 545486663
ISA List : amd64 pentium_pro+mmx pentium_pro pentium+mmx pentium
i486 i386 i86
```

[예제 4-3] sysinfo 함수 사용하기(설정)

ex4_3.c

```
01 #include <sys/systeminfo.h>
02 #include <stdlib.h>
03 #include <stdio.h>
04 #include <string.h>
05
06  int main(void) {
07      char buf[257];
08
09      if (sysinfo(SI_HOSTNAME, buf, 257) == -1) {
10          perror("sysinfo");
11          exit(1);
12      }
13      printf("Before Hostname : %s\n", buf);
14
15      strcpy(buf, "hbooks");
16      if (sysinfo(SI_SET_HOSTNAME, buf, 257) == -1) {
17          perror("sysinfo");
18          exit(1);
19      }
20
21      if (sysinfo(SI_HOSTNAME, buf, 257) == -1) {
22          perror("sysinfo");
23          exit(1);
```

호스트 이름 변경

[예제 4-3] sysinfo 함수 사용하기(설정)

```
24      }
25      printf("After Hostname : %s\n", buf);
26
27      return 0;
28  }
```

```
# ex4_3.out
Before Hostname : hanbit
After Hostname : hbooks
```

시스템 자원 정보 검색[1]

- 하드웨어에 따라 사용할 수 있는 자원들의 최댓값 검색
- 시스템 자원 정보 검색 : ~~sysinfo(2)~~ sysconf

```
#include <unistd.h>

long sysconf(int name);
```

  - 검색할 정보를 나타내는 상수를 사용해야 한다.
  - POSIX.1에서 정의한 상수

상수	설명
_SC_ARG_MAX(1)	argv[ ]와 envp[ ]를 합한 최대 크기로, 바이트 단위로 표시한다.
_SC_CHILD_MAX(2)	한 UID에 허용되는 최대 프로세스 개수를 나타낸다.
_SC_CLK_TCK(3)	초당 클록 틱 수를 나타낸다.
_SC_OPEN_MAX(5)	프로세스당 열 수 있는 최대 파일 개수를 나타낸다.
_SC_VERSION(8)	시스템이 지원하는 POSIX.1의 버전을 나타낸다.

시스템 자원 정보 검색

- SVR4에서 정의한 상수

상수	설명
_SC_PASS_MAX(9)	패스워드의 최대 길이를 나타낸다.
_SC_LOGNAME_MAX(10)	로그인명의 최대 길이를 나타낸다.
_SC_PAGESIZE(11)	시스템 메모리의 페이지 크기를 나타낸다.

- POSIX.4에서 정의한 상수

상수	설명
_SC_MEMLOCK(25)	프로세스 메모리 잠금 기능을 제공하는지 여부를 나타낸다.
_SC_MQ_OPEN_MAX(29)	한 프로세스가 열 수 있는 최대 메시지 큐 개수를 나타낸다.
_SC_SEMAPHORES(35)	시스템에서 세마포어를 지원하는지 여부를 나타낸다.

- XPG.4에서 정의한 상수

상수	설명
_SC_2_C_BIND(45)	C 언어의 바인딩 옵션을 지원하는지 여부를 알려준다.
_SC_2_C_VERSION(47)	ISO POSIX-2 표준의 버전을 나타낸다.

[예제 4-4] sysconf 함수 사용하기

ex4_4.c

```
01  #include <unistd.h>
02  #include <stdio.h>
03
04  int main(void) {
05     printf("Clock Tick : %ld\n", sysconf(_SC_CLK_TCK));
06     printf("Max Open File : %ld\n", sysconf(_SC_OPEN_MAX));
07      printf("Max Login Name Length : %ld\n", sysconf(_SC_LOGNAME_MAX));
08     // _SC_LOGIN_NAME_MAX
09      return 0;
10  }
```

```
# ex4_4.out
Clock Tick : 100
Max Open File : 256
Max Login Name Length : 8
```

시스템 자원 정보 검색[3]

파일과 디렉토리 관련 자원 검색 : fpathconf(3), pathconf(3)

```
#include <unistd.h>

long pathconf(const char *path, int name);
long fpathconf(int fildes, int name);
```

- 경로(path)나 파일기술자에 지정된 파일에 설정된 자원값이나 옵션값 리턴
- name 사용할 상수

상수	설명
_PC_LINK_MAX(1)	디렉토리 혹은 파일 하나에 가능한 최대 링크 수를 나타낸다.
_PC_NAME_MAX(4)	파일명의 최대 길이를 바이트 크기로 나타낸다.
_PC_PATH_MAX(5)	경로명의 최대 길이를 바이트 크기로 나타낸다.

[예제 4-5] pathconf 함수 사용하기

ex4_5.c

```
01  #include <unistd.h>
02  #include <stdio.h>
03
04  int main(void) {
05      printf("Link Max : %ld\n", pathconf(".", _PC_LINK_MAX));
06      printf("Name Max : %ld\n", pathconf(".", _PC_NAME_MAX));
07      printf("Path Max : %ld\n", pathconf(".", _PC_PATH_MAX));
08
09      return 0;
10  }
```

현재 디렉토리

```
# ex4_5.out
Link Max : 32767
Name Max : 255
Path Max : 1024
```

사용자 정보 검색

- 사용자 정보, 그룹정보, 로그인 기록 검색
  - /etc/passwd, /etc/shadow, /etc/group, /var/adm/utmpx
- 로그인명 검색 : getlogin(3), cuserid(3)

```
#include <unistd.h>

char *getlogin(void);
```

  - /var/adm/utmpx 파일을 검색해 현재 프로세스를 실행한 사용자의 로그인명을 리턴

```
#include <stdio.h>

char *cuserid(char *s);
```

  - 현재 프로세스의 소유자 정보로 로그인명을 찾아 리턴
- UID검색

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

uid_t getuid(void);
uid_t geteuid(void);
```

[예제 4-6] getuid, geteuid 함수 사용하기

ex4_6.c

```
01  #include <sys/types.h>
02  #include <unistd.h>
03  #include <stdio.h>
04
05  int main(void) {
06      uid_t uid, euid;
07      char *name, *cname;
08
09      uid = getuid();
10      euid = geteuid();
11
12      name = getlogin();
13      cname = cuserid(NULL);
14
15      printf("Login Name=%s, UID=%d, EUID=%d\n", name, cname,
                (int)uid, (int)euid);
16
17      return 0;
18  }
```

슈퍼유저 모드
```
su passwd root
su
./ex4_6.out
exit
```

다른 계정으로 로그인
```
su hansung2
```

```
# ex4_6.out
Login Name=root,root UID=0, EUID=0
```

특수 접근 권한 100

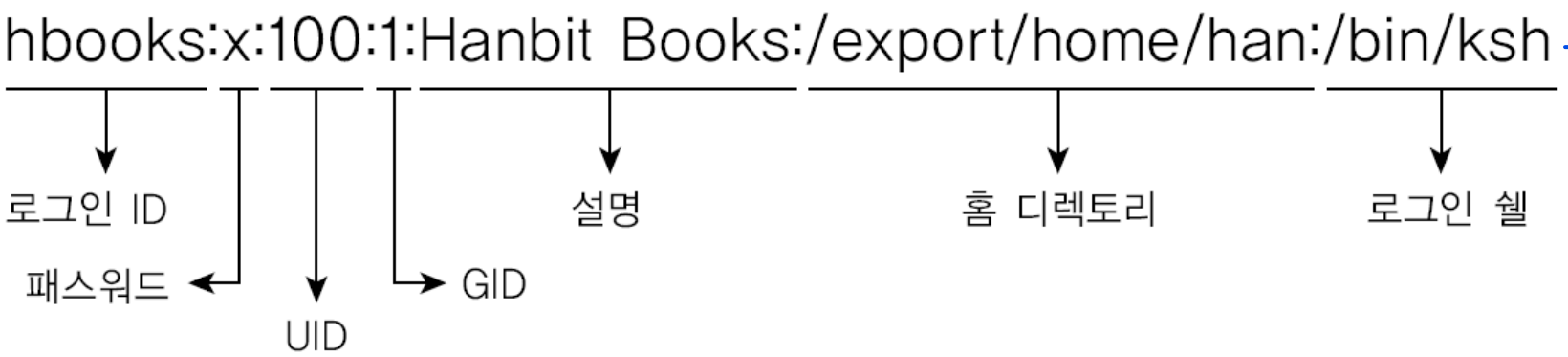
```
# chmod 4755 ex4_6.out
# ls -l ex4_6.out
-rwsr-xr-x   1 root     other        5964   1월 29일  15:11 ex4_6.out
```

setuid 설정 후 일반사용자가 이 파일을 실행하면?

패스워드 파일 검색[1]

- /etc/passwd 파일의 구조

```
# cat /etc/passwd
root:x:0:0:Super-User:/:/usr/bin/ksh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
......
hbooks:x:100:1:Hanbit Books:/export/home/han:/bin/ksh
```



[그림 4-1] 사용자 계정의 예

패스워드 파일 검색[2]

- UID로 passwd 파일 읽기 : getpwuid(3)

```
#include <pwd.h>

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

- 이름으로 passwd 파일 읽기 : getpwnam(3)

```
#include <pwd.h>

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

  - passwd 구조체

```
struct passwd {
        char   *pw_name;   이름
        char   *pw_passwd; 패스워드
        uid_t   pw_uid;    유저 id
        gid_t   pw_gid;    그룹 id
        char   *pw_age;
        char   *pw_comment;
        char   *pw_gecos;
        char   *pw_dir;    홈 디렉토리
        char   *pw_shell;  쉘
};
```

[예제 4-7] getpwuid 함수 사용하기

ex4_7.c

```
01  #include <unistd.h>
02  #include <pwd.h>
03
04  int main(void) {
05      struct passwd *pw;
06
07      pw = getpwuid(getuid());
08      printf("UID : %d\n", (int)pw->pw_uid);
09      printf("Login Name : %s\n", pw->pw_name);
10
11      return 0;
12  }
```

```
# ex4_7.out
UID : 0
Login Name : root
```

[예제 4-8] getpwnam 함수 사용하기

ex4_8.c

```
01  #include <pwd.h>
02
03  int main(void) {
04      struct passwd *pw;
05
06      pw = getpwnam("hansung");
07      printf("UID : %d\n", (int)pw->pw_uid);
08      printf("Home Directory : %s\n", pw->pw_dir);
09
10      return 0;
11  }
```

```
# ex4_8.out
UID : 100
Home Directory : /export/home/han
```

패스워드 파일 검색[3]

- /etc/passwd 파일 순차적으로 읽기

```
#include <pwd.h>

struct passwd *getpwent(void);
void setpwent(void);
void endpwent(void);
struct passwd *fgetpwent(FILE *fp);
```

해당 엔트리 1개 확인
패스워드 파일의 맨앞으로 이동
닫기(close)
고수준으로 해당 엔트리 1개 확인

[예제 4-9] getpwent 함수 사용하기 ex4_9.c

```
01  #include <pwd.h>
02
03  int main(void) {
04      struct passwd *pw;
05      int n;
07      for (n = 0; n < 3; n++) {
08          pw = getpwent();
09          printf("UID: %d, LoginName: %s\n", (int)pw->pw_uid,
                    pw->pw_name);
10      }
11
12      return 0;
13  }
```

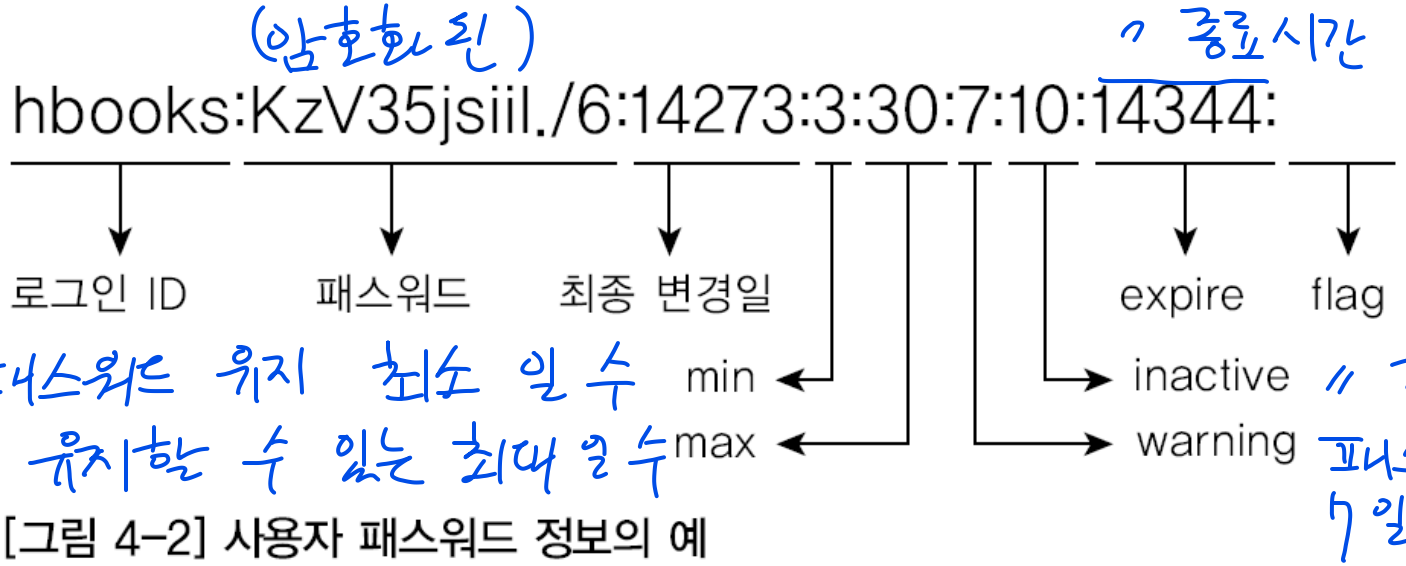
많지만 3개만

```
# ex4_9.out
UID: 0, LoginName: root
UID: 1, LoginName: daemon
UID: 2, LoginName: bin
```

섀도우 파일 검색[1]

- /etc/shadow 파일의 구조 슈퍼 유저만 접근 가능

```
# cat /etc/shadow
root:lyTy6ZkWh4RYw:13892::::::
daemon:NP:6445::::::
bin:NP:6445::::::
......
hbooks:KzV35jsiil./6:14273:3:30:7:10:14344:
```



[그림 4-2] 사용자 패스워드 정보의 예

섀도우 파일 검색[2]

/etc/shadow 파일 읽기 : getspnam(3)

```
#include <shadow.h>

struct spwd *getspnam(const char *name);
```

- spwd 구조체

```
struct spwd {
        char  *sp_namp;
        char  *sp_pwdp;
        int    sp_lstchg;
        int    sp_min;
        int    sp_max;
        int    sp_warn;
        int    sp_inact;
        int    sp_expire;
        unsigned int sp_flag;
};
```

이름
패스워드
마지막 패스워드 변경일
유지해야하는 최소 일 수
유지할 수 있는 최대 일 수
만료 경고 일 수
.
.
.

[예제 4-10] getspnam 함수 사용하기

ex4_10.c

```
01  #include <shadow.h>
02
03  int main(void) {
04      struct spwd *spw;
05
06      spw = getspnam("hansung");
07      printf("Login Name : %s\n", spw->sp_namp);
08      printf("Passwd : %s\n", spw->sp_pwdp);
09      printf("Last Change : %d\n", spw->sp_lstchg);
10
11      return 0;
12  }
```

오류날 시 %ld

```
# ex4_10.out
Login Name : hbooks
Passwd : KzV35jsiil./6
Last Change : 14273
```

섀도우 파일 검색[3]

- /etc/shadow 파일 순차적으로 읽기

```
#include <shadow.h>

struct spwd *getspent(void);
void setspent(void);
void endspent(void);
struct spwd *fgetspent(FILE *fp);
```

엔트리 1개씩 읽기
맨앞으로
close

[예제 4-11] getspent 함수 사용하기 ex4_11.c

```
01  #include <shadow.h>
02
03  int main(void) {
04      struct spwd *spw;
05      int n;
06
07      for (n = 0; n < 3; n++) {
08          spw = getspent();
09          printf("LoginName: %s, Passwd: %s\n", spw->sp_namp,
                    spw->sp_pwdp);
10      }
11
12      return 0;
13  }
```

```
# ex4_11.out
LoginName: root, Passwd: lyTy6ZkWh4RYw
LoginName: daemon, Passwd: NP
LoginName: bin, Passwd: NP
```

그룹 정보 검색

- 그룹 ID 검색하기 : getgid(2), getegid(2)

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

gid_t getgid(void);
gid_t getegid(void);
```

[예제 4-12] getid, getegid 함수 사용하기 ex4_12.c

```
01  #include <sys/types.h>
02  #include <unistd.h>
03  #include <stdio.h>
04
05  int main(void) {
06      gid_t gid, egid;
07
08      gid = getgid();
09      egid = getegid();
10
11      printf("GID=%d, EGID=%d\n", (int)gid, (int)egid);
12
13      return 0;
14  }
```

```
# ex4_12.out
GID=1, EGID=1
```

그룹 파일 검색[1]

/etc/group 파일의 구조

```
# cat /etc/group
root::0:
other::1:root
bin::2:root,daemon
sys::3:root,bin,adm
adm::4:root,daemon
uucp::5:root
......
```

- group 구조체

```
struct group {
        char   *gr_name;
        char   *gr_passwd;
        gid_t   gr_gid;
        char   **gr_mem;
};
```

이름
패스워드
그룹 id
그룹 멤버들 이름

그룹 파일 검색[2]

- /etc/group 파일 검색 : getgrnam(3), getgrgid(3)

```
#include <grp.h>

struct group *getgrnam(const char *name);
struct group *getgrgid(gid_t gid);
```

- /etc/group 파일 순차적으로 읽기

```
#include <grp.h>

struct group *getgrent(void);
void setgrent(void);
void endgrent(void);
struct group *fgetgrent(FILE *fp);
```

[예제 4-13] getgrnam 함수 사용하기

ex4_13.c

```
01  #include <grp.h>
02
03  int main(void) {
04      struct group *grp;
05      int n;
06
07      grp = getgrnam("adm");
08      printf("Group Name : %s\n", grp->gr_name);
09      printf("GID : %d\n", (int)grp->gr_gid);
10
11      n = 0;
12      printf("Members : ");
13      while (grp->gr_mem[n] != NULL)
14          printf("%s ", grp->gr_mem[n++]);
15      printf("\n");
16
17      return 0;
18  }
```

없을시 gdm으로해봐

```
# ex4_13.out
Group Name : adm
GID : 4
Members : root daemon
```

[예제 4-14] getgrent 함수 사용하기

ex4_14.c

```
01  #include <grp.h>
02
03  int main(void) {
04      struct group *grp;
05      int n,m;
06
07      for (n = 0; n < 3; n++) {
08          grp = getgrent();
09          printf("GroupName: %s, GID: %d ", grp->gr_name,
                    (int)grp->gr_gid);
10
11          m = 0;
12          printf("Members : ");
13          while (grp->gr_mem[m] != NULL)
14              printf("%s ", grp->gr_mem[m++]);
15          printf("\n");
16      }
17
18      return 0;
19  }
```

```
# ex4_14.out
GroupName: root, GID: 0 Members : 멤버 없다
GroupName: other, GID: 1 Members : root
GroupName: bin, GID: 2 Members : root daemon
```

로그인 기록 검색[1]

- who 명령 : 현재 시스템에 로그인하고 있는 사용자 정보
- last 명령 : 시스템의 부팅 시간 정보와 사용자 로그인 기록 정보
- utmpx 구조체

```
struct utmpx {
    char      ut_user[32];        /* 사용자 로그인명 */
    char      ut_id[4];           /* inittab id */
    char      ut_line[32];        /* 로그인한 장치이름 */
    pid_t     ut_pid;             /* 실행중인 프로세스 PID*/
    short     ut_type;            /* 현재 읽어온 항목의 종류 */
    struct    exit_status ut_exit;/* 프로세스 종료 상태 코드 */
    struct    timeval ut_tv;      /* 해당정보를 변경한 시간 */
    int       ut_session;         /* 세션 번호 */
    int       pad[5];             /* 예약 영역 */
    short     ut_syslen;          /* ut_host의 크기 */

    char      ut_host[257];       /* 원격호스트명 */
};
```

로그인 기록 검색[2]

ut_type : 현재 읽어온 항목의 종류

- EMPTY(0) : 비어 있는 항목이다.
- RUN_LVL(1) : 시스템의 런레벨이 변경되었음을 나타낸다. 바뀐 런레벨은 ut_id에 저장된다.
- BOOT_TIME(2) : 시스템 부팅 정보를 나타낸다. 부팅 시간은 ut_time에 저장된다.
- OLD_TIME(3) : date 명령으로 시스템 시간이 변경되었음을 나타낸다. 변경되기 전의 시간을 저장한다.
- NEW_TIME(4) : date 명령으로 시스템 시간이 변경되었음을 나타낸다. 변경된 시간을 저장한다.
- INIT_PROCESS(5) : init에 의해 생성된 프로세스임을 나타낸다. 프로세스명은 ut_name에 저장하고 프로세스 ID는 ut_pid에 저장한다.
- LOGIN_PROCESS(6) : 사용자가 로그인하기를 기다리는 getty 프로세스를 나타낸다
- USER_PROCESS(7) : 사용자 프로세스를 나타낸다.
- DEAD_PROCESS(8) : 종료한 프로세스를 나타낸다.
- ACCOUNTING(9) : 로그인 정보를 기록한 것임을 나타낸다.
- DOWN_TIME(10) : 시스템을 다운시킨 시간을 나타낸다. ut_type이 가질 수 있는 가장 큰 값이다.

로그인 기록 검색[3]

- /var/adm/utmpx 파일 순차적으로 읽기

```
#include <utmpx.h>

struct utmpx *getutxent(void);
void setutxent(void);
void endutxent(void);
int utmpxname(const char *file);
```

[예제 4-15] getutxent 함수 사용하기 ex4_15.c

```
#include <sys/types.h><utmpx.h><stdio.h>
05  int main(void) {
06      struct utmpx *utx;
07
08      printf("LoginName  Line\n");
09      printf("---------------\n");
10
11      while ((utx=getutxent()) != NULL) {
12          if (utx->ut_type != USER_PROCESS)
13              continue;
14
15          printf("%s       %s\n", utx->ut_user, utx->ut_line);
16      }
17
18      return 0;
19  }
```

```
# ex4_15.out
LoginName  Line
---------------
root       console
root       pts/3
root       pts/5
root       pts/4
```

시간 관리 함수[1]

유닉스 시스템에서 시간관리

- 1970년 1월 1일 0시 0분 0초(UTC)를 기준으로 현재까지 경과한 시간을 초 단위로 저장하고 이를 기준으로 시간 정보 관리

초 단위로 현재 시간 정보 얻기 : time(2)

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

time_t time(time_t *tloc);
```

[예제 4-16] time 함수 사용하기 ex4_16.c

```
01  #include <sys/types.h>
02  #include <time.h>
03  #include <stdio.h>
04
05  int main(void) {
06      time_t tt;
07
08      time(&tt);
09      printf("Time(sec) : %d\n", (int)tt);
10
11      return 0;
12  }
```

```
# ex4_16.out
Time(sec) : 1233361205
```

시간 관리 함수[2]

마이크로 초 단위로 시간 정보얻기 : gettimeofday(3)

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

int gettimeofday(struct timeval *tp, void *tzp);
int settimeofday(struct timeval *tp, void *tzp);
```

- timeval 구조체

```
struct timeval {
    time_t      tv_sec; /* 초 */
    suseconds_t tv_usec; /* 마이크로 초 */
};
```

[예제 4-17] gettimeofday 함수 사용하기 ex4_17.c

```
..
04  int main(void) {
05      struct timeval tv;
06
07      gettimeofday(&tv, NULL);
08      printf("Time(sec) : %d\n", (int)tv.tv_sec);
09      printf("Time(micro-sec) : %d\n", (int)tv.tv_usec);
10
11      return 0;
12  }
```

```
# ex4_17.out
Time(sec) : 1233362365
Time(micro-sec) : 670913
```

시간 관리 함수[3]

시간대 정보 : tzset(3)

```
#include <time.h>

void tzset(void);
```

- 현재 지역의 시간대로 시간을 설정
- 이 함수를 호출하면 전역변수 4개에 정보를 설정
  - timezone : UTC와 지역 시간대와 시차를 초 단위로 저장
  - altzone : UTC와 일광절약제 등으로 보정된 지역시간대와의 시차를 초 단위로 저장
  - daylight : 일광절약제를 시행하면 0이 아니고, 아니면 0
  - tzname : 지역시간대와 보정된 시간대명을 약어로 저장

[예제 4-18] tzset 함수 사용하기 ex4_18.c

```
01  #include <time.h>
02  #include <stdio.h>
03
04  int main(void) {
05      tzset();
06
07      printf("Timezone : %d\n", (int)timezone);
08      // printf("Altzone : %d\n", (int)altzone);
09      printf("Daylight : %d\n", daylight);
10      printf("TZname[0] : %s\n", tzname[0]);
11      printf("TZname[1] : %s\n", tzname[1]);
12
13      return 0;
14  }
```

```
# ex4_18.out
Timezone : -32400
Altzone : -36000
Daylight : 1
TZname[0] : KST
TZname[1] : KDT
```

UTC와 9시간(32,400초) 시차가 발생

시간의 형태 변환

초 단위 시간 정보 분해 : gmtime(3), localtime(3)

```
#include <time.h>

struct tm *localtime(const time_t *clock);
struct tm *gmtime(const time_t *clock);
```

- 초를 인자로 받아 tm구조 리턴, gmtime은 UTC기준, localtime은 지역시간대 기준

초 단위 시간으로 역산 : mktime(3)

```
#include <time.h>

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

- tm구조체

```
struct tm {
    int tm_sec;
    int tm_min;
    int tm_hour;
    int tm_mday;
    int tm_mon;
    int tm_year;
    int tm_wday;
    int tm_yday;
    int tm_isdst;
};
```

tm_mon(월) : 0(1월)~11(12월)
tm_year(연도) : 년도 – 1900
tm_wday(요일) : 0(일)~6(토)
tm_isdst(일광절약제) : 1(시행)

[예제 4-19] gmtime, localtime 함수 사용하기

ex4_19.c

```
01 #include <time.h>
02 #include <stdio.h>
03
04  int main(void) {
05      struct tm *tm;
06      time_t t;
07
08      time(&t);
09      printf("Time(sec) : %d\n", (int)t);
10
11      tm = gmtime(&t);
12      printf("GMTIME=Y:%d ", tm->tm_year);
13      printf("M:%d ", tm->tm_mon);
     printf("D:%d ", tm->tm_mday);
15      printf("H:%d ", tm->tm_hour);
16      printf("M:%d ", tm->tm_min);
17      printf("S:%d\n", tm->tm_sec);
18
19      tm = localtime(&t);
20      printf("LOCALTIME=Y:%d ", tm->tm_year);
21      printf("M:%d ", tm->tm_mon);
22      printf("D:%d ", tm->tm_mday);
```

[예제 4-19] gmtime, localtime 함수 사용하기

```
23      printf("H:%d ", tm->tm_hour);
24      printf("M:%d ", tm->tm_min);
25      printf("S:%d\n", tm->tm_sec);
26
27      return 0;
28  }
```

```
# ex4_19.out
Time(sec) : 1233369331
GMTIME=Y:109 M:0 D:31 H:2 M:35 S:31
LOCALTIME=Y:109 M:0 D:31 H:11 M:35 S:31
```

연도가 109? +1900
어떻게 해석해야하나?

[예제 4-20] mktime 함수 사용하기

ex4_20.c

```
01  #include <time.h>
02  #include <stdio.h>
03
04  int main(void) {
05      struct tm tm;
06      time_t t;
07
08      time(&t);
09      printf("Current Time(sec) : %d\n", (int)t);
10
11      tm.tm_year = 109;
12      tm.tm_mon = 11;
13      tm.tm_mday = 31;
14      tm.tm_hour = 12;
15      tm.tm_min = 30;
16      tm.tm_sec = 0;
17
18      t = mktime(&tm);
19      printf("2009/12/31 12:30:00 Time(sec) : %d\n", (int)t);
20
21      return 0;
22  }
```

```
# ex4_20.out
Current Time(sec) : 1233370219
2009/12/31 12:30:00 Time(sec) : 1262226600
```

형식 지정 시간 출력[1]

- 초 단위 시간을 변환해 출력하기: ctime(3)

```
#include <time.h>

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

[예제 4-21] ctime 함수 사용하기 ex4_21.c

```
01  #include <time.h>
02  #include <stdio.h>
03
04  int main(void) {
05      time_t t;
06
07      time(&t);
08
09      printf("Time(sec) : %d\n", (int)t);
10      printf("Time(date) : %s\n", ctime(&t));
11
12      return 0;
13  }
```

```
# ex4_21.out
Time(sec) : 1233370759
Time(date) : Sat Jan 31 11:59:19 2009
```

형식 지정 시간 출력[2]

- tm 구조체 시간을 변환해 출력하기: asctime(3) = ctime

```
#include <time.h>

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

[예제 4-22] asctime 함수 사용하기 ex4_22.c

```
01  #include <time.h>
02  #include <stdio.h>
03
04  int main(void) {
05      struct tm *tm;
06      time_t t;
07
08      time(&t);
09      tm = localtime(&t);
10
11      printf("Time(sec) : %d\n", (int)t);
12      printf("Time(date) : %s\n", asctime(tm));
13
14      return 0;
15  }
```

```
# ex4_22.out
Time(sec) : 1233371061
Time(date) : Sat Jan 31 12:04:21 2009
```

형식 지정 시간 출력[3]

- 출력 형식 기호 사용 : strftime(3)

```
#include <time.h>

size_t strftime(char *restrict s, size_t maxsize,
const char *restrict format, const struct tm *restrict timeptr);
```

- s : 출력할 시간 정보를 저장할 배열 주소
- maxsize : s의 크기
- format : 출력 형식
- timeptr : 출력할 시간정보를 저장한 구조체 주소

형식 지정 시간 출력[3]

지정자	기능	지정자	기능
%a	지역 시간대의 요일명 약자	%A	지역 시간대의 요일명
%b	지역 시간대의 월 이름 약자	%B	지역 시간대의 월 이름
%c	지역 시간대에 적합한 날짜와 시간 표현	%C	date 명령의 결과와 같은 형태로 날짜와 시간 표현
%d	날짜(0~31)	%D	날짜(%m/%d/%y)
%e	날짜(0~31). 한 자리 수는 앞에 공백 추가	%F	%Y-%m-%d 형태로 표현
%g	년도(00~99)	%G	년도(0000~9999)
%h	지역 시간대 월 이름 약자	%j	1년 중 일 수(001~365)
%H	24시간 기준 시간(00~23)	%I	12시간 기준 시간(01~12)
%k	24시간 기준 시간(00~23), 한 자리 수는 앞에 공백 추가	%l	12시간 기준 시간(01~12), 한 자리 수는 앞에 공백 추가
%m	월(01~12)	%M	분(00~59)
%p	지역 시간대 a.m, p.m	%r	%p와 함께 12시간 표시
%R	%H:%M 형태로 시간 표시	%T	%H:%M:%S 형태로 시간 표시
%S	초(00~60)		

형식 지정 시간 출력[3]

%n	개행	%t	탭 추가
%U	연중 주간 수 표시(00~53)	%V	ISO 8601 표준으로 연중 주간 수 표시(01~53)
%w	요일(0~6, 0을 일요일)	%W	연중 주간 수 표시(00~53), 1월 첫 월요일이 01주, 그 이전은 00주로 표시
%x	지역 시간대에 적합한 날짜 표시	%X	지역 시간대에 적합한 시간 표시
%y	년도(00~99)	%Y	네 자리 수 년도
%z	UTC와의 시차 표시	%Z	시간대명 약자

[예제 4-23] strftime 함수 사용하기

ex4_23.c

```
01  #include <time.h>
02  #include <stdio.h>
03
04  char *output[] = {
05      "%x %X",
06      "%G년, %m월 %d일 %U주 %H:%M",
07      "%r"
08  };
09
10  int main(void) {
11      struct tm *tm;
12      int n;
13      time_t t;
14      char buf[257];
15
16      time(&t);
17      tm = localtime(&t);
18
19      for (n = 0; n < 3; n++) {
20          strftime(buf, sizeof(buf), output[n], tm);
21          printf("%s = %s\n", output[n], buf);
22      }
23
24      return 0;
25  }
```

```
# ex4-23.out
%x %X = 01/31/09 12:43:12
%G년 %m월 %d일 %U주 %H:%M = 2009년 01월 31일 04주 12:43
%r = 12:43:12 PM
```

Thank You !

IT CookBook, 유닉스 시스템 프로그래밍

한빛미디어 Hanbit Media, Inc.