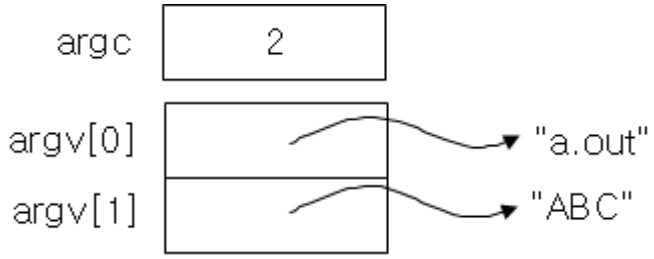
리눅스 프로그램 환경

2007. 11. 28

안 효 창

명령라인 인수 사용하기

```
[jkim@localhost program]$ ls -l
⋮
[jkim@localhost program]$ cat -n test.c
⋮
[jkim@localhost program]$

main(int argc, char *argv[])
{
⋮
}

[jkim@localhost program]$ a.out ABC
```



명령라인 인수 사용하기

20-1

```
#include <stdio.h>

/* 명령라인 인수를 가져온다. argc에는 인수의 개수
      를, argv에는 인수의 내용을 저장한다. */
main(int argc, char *argv[])
{
   int i;

   printf("argc : %d₩n", argc);   /* 인수의 개수 출력
      */

   /* 인수의 내용 출력 */
   for(i=0; i<argc; i++)
      printf("argv[%d] : %s₩n", i, argv[i]);

   exit(0);
}
```

20-2

```
#include <stdio.h>
#include <stdlib.h>  /* atoi 함수를 정의하는 헤더
      파일 */
int max(int num1, int num2);
main(int argc, char *argv[])
{
   /* 인수를 잘못 주어 실행시켰을 때 */
   if (argc != 3) {
      printf("Usage : a.out number1 number2₩n");
      exit(1);
   }
 /* atoi는 정수 형태의 문자열을 정수로 변환 함수 */
   printf("max number : %d₩n", max(atoi(argv[1]),
      atoi(argv[2])));
   exit(0);
}
int max(int num1, int num2)
{
   if (num1 > num2)
      return num1;
   else
      return num2;
}
```

명령라인 인수 사용하기

- 20-3

```
#include <stdio.h>

main(int argc, char *argv[])
{
   int i;
   /* 첫 번째 인수를 제외한 모든 인수들에 대해 */
   for (i=1; i<argc; i++) {
      /* 명령라인 인수의 첫 글자가 '-'면 옵션 */
      if(argv[i][0] == '-')
         printf("option : %s₩n", argv[i]+1);
      /* 그렇지 않으면 인수 */
      else
         printf("argument : %s₩n", argv[i]);
   }
}
```

명령라인 인수 사용하기

getopt 함수
기능
명령라인 인수를 분석한다.
기본형
int getopt(int argc, char * const argv[], const char *optstring);
extern char *optarg;
extern int optind, opterr, optopt;
argc : 명령라인 인수의 개수
argv : 명령라인 인수의 내용
optstring : 옵션 목록
optarg : 인수를 필요로 하는 옵션을 처리할 때 인수를 가리킴
optind : 다음에 처리할 인수의 argv 첨자
opterr : 오류 메시지 출력 여부를 결정하는 변수로 0으로 설정되면 출력하지 않음
optopt : 인식되지 않는 옵션
반환값
성공 : 옵션
인식되지 않은 옵션 : 물음표(?)
옵션이 없으면 : -1
헤더 파일
<unistd.h>

명령라인 인수 사용하기

```
getopt(argc, argv, "lai");
```

```
[jkim@localhost program]$ a.out -l -a
```

- 첫 호출에서는 l을 반환하고, 다음 호출은 a를 반환

```
[jkim@localhost program]$ a.out -k
```

- optstring에 지정하지 않은 옵션을 주면 물음표(?)를 반환하고 optopt에 인식되지 않은 옵션인 k가 저장

- 인수가 필요한 옵션의 경우에는 콜론(:)을 추가

```
getopt(argc, argv, "lf:ai");
```

```
[jkim@localhost program]$ a.out -a -f test
```

- getopt가 f를 반환할 때 optarg가 인수인 'test'를 가리킴

명령라인 인수 사용하기

- optind는 다음에 처리할 인수의 argv 첨자로 초기에는 1로 설정

[jkim@localhost program]$ a.out -l -a

  - getopt를 호출하기 전에 optind는 1인데 getopt가 l 옵션을 반환하면 optind는 '-a'를 의미하는 argv[2]의 첨자인 2가 된다.

- getopt의 오류메시지를 출력하지 않으려면 opterr을 0으로 설정

명령라인 인수 사용하기

20-4

```
#include <stdio.h>
#include <unistd.h>   /* getopt 함수를 정의하는 헤더 파일 */

main(int argc, char *argv[])
{
  int opt;
  /* 명령라인의 l, f, a, i, 옵션을 처리 */
  while ((opt = getopt(argc, argv, "lf:ai")) != -1) {
    switch(opt) {
      /* l, a, i 옵션이면 옵션을 출력 */
      case 'l':
      case 'a':
      case 'i':
        printf("option : %c₩n", opt);
        break;
      /* f 옵션이면 옵션과 인수를 출력 */
      case 'f':
        printf("option %c's argument : %s₩n", opt, optarg);
        break;
      /* 인식되지 않는 옵션 출력 */
      default:
        printf("unknown option : %c₩n", optopt);
    }
  }
  exit(0);
}
```

20-5

```
#include <stdio.h>
#include <unistd.h>   /* getopt 함수를 정의하는 헤더 파일 */

main(int argc, char *argv[])
{
  int opt;
  while ((opt = getopt(argc, argv, "lf:ai")) != -1) {
    switch(opt) {
      case 'l':
      case 'a':
      case 'i':
        printf("option : %c₩n", opt);
        break;
      case 'f':
        printf("option %c's argument : %s₩n", opt, optarg);
        break;
      case '?':
        printf("unknown option : %c₩n", optopt);
    }
  }
  /* 처리되지 않은 인수들을 출력, optind에 다음에 처리해야
     할 인수의 argv 첨자 저장 */
  for (; optind<argc; optind++)
    printf("argument : %s₩n", argv[optind]);
  exit(0);
}
```

라인 번호와 함께 파일 내용 출력

- 20-6

```
#include <stdio.h>
void output_file(FILE *fp);
main(int argc, char *argv[])
{
   FILE *fp;
   if (argc == 1) {    /* 인수 잘못주어 실행하
      면 */
      printf("Usage : a.out filename ...₩n");
      exit(1);
   }
/* 여러 개의 파일을 처리할 수 있도록 반복 */
while (*++argv) {
printf("₩n[filename : %s]₩n₩n", *argv);
    /* 명령라인 인수에 해당되는 파일을 염 */
if((fp = fopen(*argv, "r")) == NULL) {
        perror("fopen failed");
        exit(2);
      }
output_file(fp);
      fclose(fp);
   }
   exit(0);
}

void output_file(FILE *fp)
{
   int ch;

   /* 한 글자씩 읽어서 화면으로 출력 */
   while((ch=getc(fp)) != EOF)
      putc(ch, stdout);
}
```

- 셸 명령어인 cat과 같은 동작

라인 번호와 함께 파일 내용 출력

- [프로그램 20-7], [프로그램 20-8]
- 출력 내용 앞에 라인 번호가 출력
- 셸 명령어인 cat -n과 같은 동작

환경 변수

- 실행 중인 모든 프로그램에는 기본적으로 설정된 정보가 있는데 이를 저장한 변수
- 환경 변수를 확인하는 셸 명령어는 printenv
  - 변수이름=변수값

명령라인의 envp 인수

```
main(int argc, char *argv[], char *envp[])
{
⋮
}
```

- 20-9

```
#include <stdio.h>
#include <stdlib.h>

/* envp에 환경 변수 정보 저장 */
main(int argc, char *argv[], char *envp[])
{
   while(*envp)
      printf("%s₩n", *envp++);
}
```

environ 변수

environ 변수
기능
　　환경 변수 정보를 저장한다.
기본형
　　extern char **environ;
헤더 파일
　　<stdlib.h>

- 20-10

```
#include <stdio.h>
#include <stdlib.h>   /* environ 변수를 선언하는 헤더 파일 */

/* environ에 환경 변수 정보 저장 */
extern char **environ;

main()
{
  while(*environ)
    printf("%s₩n", *environ++);

}
```

환경 변수 관련 함수

```
char *getenv(const char *name);
int putenv(const char *string);
int setenv(const char *name, const char *value, int overwrite);
void unsetenv(const char *name);
```

getenv 함수

기능

환경 변수 값을 가져온다.

기본형

char *getenv(const char *name);

name : 알고자 하는 환경 변수 이름

반환값

성공 : 환경 변수 값

실패 : NULL

헤더 파일

<stdlib.h>

환경 변수 관련 함수

- 20-11

```
#include <stdio.h>
#include <stdlib.h>   /* getenv 함수를 정의하는 헤더 파일 */

main()
{
   char *home_dir, *work_dir;

   /* getenv를 이용해 HOME에 대한 환경 변수 값 얻어 옴 */
   if((home_dir=getenv("HOME")) != NULL)
      printf("home directory : %s₩n", home_dir);

   /* getenv를 이용해 PWD에 대한 환경 변수 값 얻어 옴 */
   if((work_dir = getenv("PWD")) != NULL)
      printf("working directory : %s₩n", work_dir);

   /* LINUX라는 환경 변수가 없으므로 NULL을 반환 */
   if((tmp = getenv("LINUX")) != NULL)
      printf("temp : %s₩n", tmp);
}
```

환경 변수 관련 함수

putenv 함수

기능

환경 변수 값을 변경한다.

기본형

int putenv(const char *string);

string : 변경하고자 하는 환경 변수 정보로 '변수이름=변수값' 형식

반환값

성공 : 0

실패 : -1

헤더 파일

<stdlib.h>

환경 변수 관련 함수

- 20-12

```
#include <stdio.h>
#include <stdlib.h>   /* getenv, putenv 함수를 정의
      하는 헤더 파일 */

main()
{
   char *home_dir;

   /* HOME 환경 변수 값을 얻어 와서 출력 */
   if((home_dir=getenv("HOME")) != NULL)
      printf("home directory : %s₩n", home_dir);

   /* HOME 환경 변수 값을 /home/jkim/book/linux
      로 설정 */
   putenv("HOME=/home/jkim/book/linux");

   if((home_dir=getenv("HOME")) != NULL)
      printf("home directory : %s₩n", home_dir);
}
```

- [프로그램 20-13]

환경 변수 관련 함수

setenv 함수

기능

환경 변수 값을 변경한다.

기본형

int setenv(const char *name, const char *value, int overwrite);

name : 변경하고자 하는 환경 변수 이름

value : 새롭게 설정할 환경 변수 값

overwrite : 변수가 이미 존재할 때 변경할지를 결정하는 인수로 0이면 변경하지 않음

반환값

성공 : 0

실패 : -1

헤더 파일

<stdlib.h>

환경 변수 관련 함수

- 20-14

```
#include <stdio.h>
#include <stdlib.h>   /* setenv, getenv 함수를 정의하는 헤더 파일 */

main()
{
   char *value;

   /* overwrite가 0이므로 이미 HOME이 존재하면 변경하지 않음 */
   setenv("HOME", "/home/jkim/book/linux", 0);
   value = getenv("HOME");
   printf("[setenv overwrite : 0] HOME : %s₩n", value);

   /* overwrite가 1이므로 이미 HOME이 존재해도 값을 변경 */
   setenv("HOME", "/home/jkim/book/linux", 1);
   value = getenv("HOME");
   printf("[setenv overwrite : 1] HOME : %s₩n", value);

   exit(0);
}
```

환경 변수 관련 함수

- unsetenv 함수
  기능
  환경 변수를 삭제한다.
  기본형
  void unsetenv(const char *name);
  name : 삭제하고자 하는 환경 변수 이름
  반환값
  없음
  헤더 파일
  <stdlib.h>

환경 변수 관련 함수

- 20-15

```
#include <stdio.h>
#include <stdlib.h>   /* getenv, unsetenv 함수를 정의하는 헤더 파일 */

main()
{
   char *value;
   if((value=getenv("HOME")) != NULL)
      printf("HOME : %s₩n", value);
   else
      printf("HOME : no value₩n");
   /* HOME 환경 변수를 삭제 */
   unsetenv("HOME");
   /* HOME 환경 변수 없으므로 NULL 반환 */
   if((value=getenv("HOME")) != NULL)
      printf("HOME : %s₩n", value);
   else
      printf("HOME : no value₩n");
   exit(0);
}
```

사용자에 대한 이해

- 사용자 ID(user ID)

getuid, geteuid 함수

기능

getuid는 실제 사용자 ID를 반환하고, geteuid는 유효 사용자 ID를 반환한다.

기본형

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

반환값

성공 : 사용자 ID를 반환

실패 : 하지 않음

헤더 파일

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

그룹에 대한 이해

- 그룹 ID(group ID)

getgid, getegid 함수

기능

getgid는 실제 그룹 ID를 반환하고, getegid는 유효 그룹 ID를 반환한다.

기본형

```
uid_t getgid(void);
uid_t getegid(void);
```

반환값

성공 : 그룹 ID를 반환

실패 : 하지 않음

헤더 파일

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

그룹에 대한 이해

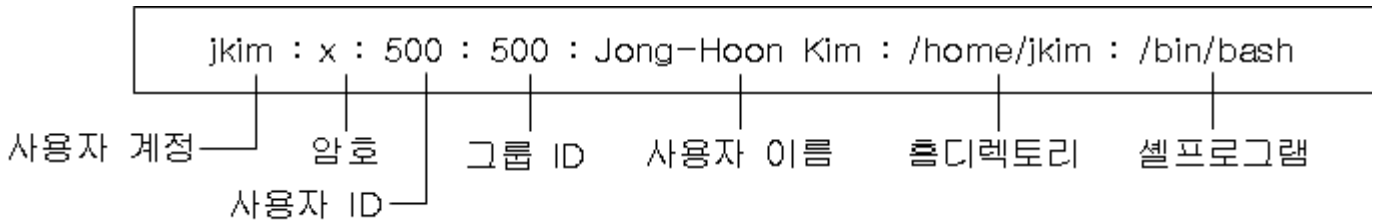
- 20-16

```
#include <stdio.h>
#include <string.h>
#include <pwd.h>
#include <grp.h>
#include <unistd.h>
#include <sys/types.h>

main()
{
  uid_t uid;
  gid_t gid;
  struct passwd *pw;
  struct group *pg;
  int i;

  uid = getuid();  /* 사용자 ID 얻어 옴 */
  pw = getpwuid(uid);  /* 패스워드 파일로부터 사용
     자 정보 얻어 옴 */
  gid = getgid();  /* 그룹 ID 얻어 옴 */
  pg = getgrgid(gid);  /* 그룹 파일로부터 그룹 정보
     얻어 옴 */
  printf("uid=%d(%s) ", uid, pw->pw_name);  /* 사
     용자 ID와 사용자 계정 출력 */
  printf("gid=%d(%s) groups=", gid, pg-
     >gr_name);  /* 그룹 ID와 그룹 이름 출력 */
  /* 그룹 파일로부터 모든 그룹 정보를 차례로 얻어
     옴 */
  while (pg = getgrent()) {
    /* 해당 그룹에 속한 모든 사용자 계정에 대해 */
    for (i=0; pg->gr_mem[i]!=NULL; i++) {
      /* 프로그램의 사용자 계정(pw->pw_name)과
     pg가 가리키는 그룹에 속한
        사용자 계정이 같다는 것은 현 사용자가 pg
     그룹에 속해있다는 것을
        의미한다. 조건이 참이면 즉, 현 사용자 계정
     이 같으면
        pg가 가리키는 그룹 이름을 출력한다. */
      if (!strcmp(pw->pw_name, pg->gr_mem[i]))
        printf("%d(%s) ", pg->gr_gid, pg-
     >gr_name);
    }
  }
  printf("₩n");
  exit(0);
}
```

사용자 정보 알아내기

- /etc/passwd 파일(패스워드 파일)

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
⋮
jkim:x:500:500:Jong-Hoon Kim:/home/jkim:/bin/bash
⋮
$
```



사용자 정보 알아내기

- 사용자에 대한 정보

```
struct passwd
{
char *pw_name; /* 사용자 계정 */
char *pw_passwd; /* 암호 */
uid_t pw_uid; /* 사용자 ID */
gid_t pw_gid; /* 그룹 ID */
char *pw_gecos; /* 사용자 이름 */
char *pw_dir; /* 홈 디렉토리 */
char *pw_shell; /* 셸 프로그램 */
};
```

사용자 정보 알아내기

getpwuid, getpwnam 함수

기능

패스워드 파일로부터 지정한 사용자에 대한 정보를 얻어온다.

기본형

```
struct passwd *getpwuid(uid_t uid);
struct passwd *getpwnam(const char * name);
```

uid : 정보를 얻고자 하는 사용자 ID

name : 정보를 얻고자 하는 사용자 계정

반환값

성공 : 사용자 정보에 대한 포인터

실패 : NULL

헤더 파일

```
<pwd.h>
<sys/types.h>
```

사용자 정보 알아내기

- 20-17

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

main()
{
   uid_t uid;
   struct passwd *pw;

   /* 실제 사용자 ID를 얻어 옴 */
   uid = getuid();
   /* uid에 대한 사용자 정보를 얻어 옴 */
   pw = getpwuid(uid);
   /* pw->pw_name는 사용자 계정, pw->pw_uid은 사용자 ID, pw->pw_gid는
     사용자가 속한 그룹 ID, pw->pw_dir는 사용자의 홈 디렉토리를 의미 */
   printf("name:%s, uid:%d, gid:%d, home:%s₩n", pw->pw_name, pw->pw_uid, pw->pw_gid, pw-
      >pw_dir);

   exit(0);
}
```

사용자 정보 알아내기

getlogin 함수
기능
사용자 계정을 얻어온다.
기본형
char *getlogin(void);
반환값
성공 : 사용자 계정
실패 : NULL
헤더 파일
<unistd.h>

getpwent 함수
기능
패스워드 파일로부터 한 사용자 정보를 얻어온다.
기본형
struct passwd *getpwent(void);
반환값
성공 : 한 사용자 정보에 대한 포인터
실패 : NULL
헤더 파일
<pwd.h>
<sys/types.h>

사용자 정보 알아내기

setpwent 함수
기능
패스워드 파일의 시작 지점으로 돌아가게 한다.
기본형
void setpwent(void);
반환값
없음
헤더 파일
<pwd.h>
<sys/types.h>

endpwent 함수
기능
패스워드 파일을 닫는다.
기본형
void endpwent(void);
반환값
없음
헤더 파일
<pwd.h>
<sys/types.h>

사용자 정보 알아내기

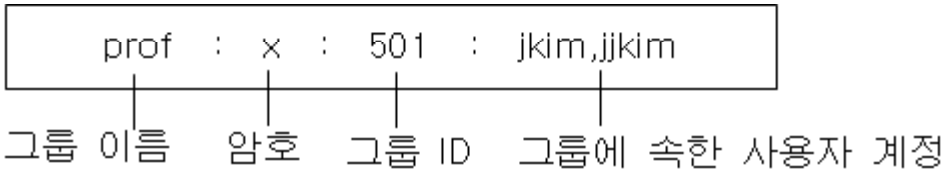
- 20-18

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

main()
{
  struct passwd *pw;

  /* 패스워드 파일로부터 사용자 정보를 차례로 읽어 출력 */
  while (pw = getpwent()) {
    /* pw->pw_name는 사용자 계정, pw->pw_uid은 사용자 ID,
       pw->pw_dir는 사용자의 홈 디렉토리를 의미 */
    printf("name:%s, uid:%d, home:%s₩n", pw->pw_name, pw->pw_uid, pw->pw_dir);
  }
  /* 사용이 끝난 패스워드 파일을 닫음 */
  endpwent();
  exit(0);
}
```

그룹 정보 알아내기

- 그룹에 대한 정보
  - /etc/group 파일(그룹 파일)

```
$ cat /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
⋮
prof:x:501:jkim,jjkim
⋮
$
```



그룹 정보 알아내기

- 그룹에 대한 정보

```
struct group
{
char *gr_name; /* 그룹 이름 */
char *gr_passwd; /* 암호 */
gid_t gr_gid; /* 그룹 ID */
char **gr_mem; /* 그룹에 속한 사용자 계정 */
};
```

그룹 정보 알아내기

- getgrent 함수
  기능
    그룹 파일로부터 한 그룹 정보를 얻어온다.
  기본형
    struct group *getgrent(void);
  반환값
    성공 : 한 그룹 정보에 대한 포인터
    실패 : NULL
  헤더 파일
    <grp.h>
    <sys/types.h>

- setgrent, endgrent 함수
  기능
    setgrent는 그룹 파일의 시작 지점으로 돌아가게 하고, endgrent는 그룹 파일을 닫는다.
  기본형
    void setgrent(void);
    void endgrent(void);
  반환값
    없음
  헤더 파일
    <grp.h>
    <sys/types.h>

그룹 정보 알아내기

- 20-19

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

main()
{
   struct group *pg;
   int i;

   /* 그룹 파일로부터 그룹 정보를 차례로 읽어 출력 */
   while (pg = getgrent()) {
      printf("group name: %s₩n   member: ", pg->gr_name);
      /* 그룹에 속한 사용자 계정 전부를 출력 */
      for (i=0; pg->gr_mem[i]!=NULL; i++)
         printf("%s ", pg->gr_mem[i]);
      printf("₩n₩n");
   }
   /* 사용이 끝난 그룹 파일을 닫음 */
   endgrent();
   exit(0);
}
```

그룹 정보 알아내기

getgrgid, getgrnam 함수
기능
그룹 파일로부터 지정한 그룹에 대한 정보를 얻어온다.
기본형
struct group *getgrgid(gid_t gid);
struct group *getgrnam(const char *name);
gid : 정보를 얻고자 하는 그룹 ID
name : 정보를 얻고자 하는 그룹 이름
반환값
성공 : 그룹 정보에 대한 포인터
실패 : NULL
헤더 파일
<grp.h>
<sys/types.h>

그룹 정보 알아내기

- 20-20

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

main()
{
   gid_t gid;
   struct group *pg;
   int i;

   /* 그룹 ID를 얻음 */
   gid = getgid();
   /* 그룹 정보를 얻음 */
   pg = getgrgid(gid);
   printf("group name : %s₩n", pg->gr_name);  /* pg->gr_name은 그룹의 이름 */
   exit(0);
}
```

사용자와 그룹 정보 알아내는 프로그램

- 20-21

```
#include <stdio.h>
#include <string.h>
#include <pwd.h>
#include <grp.h>
#include <unistd.h>
#include <sys/types.h>
main()
{
   uid_t uid;
   gid_t gid;
   struct passwd *pw;
   struct group *pg;
   int i;
   uid = getuid();  /* 사용자 ID 얻어 옴 */
   pw = getpwuid(uid);  /* 패스워드 파일로부터 사용
      자 정보 얻어 옴 */
   gid = getgid();  /* 그룹 ID 얻어 옴 */
   pg = getgrgid(gid);  /* 그룹 파일로부터 그룹 정보
      얻어 옴 */
printf("uid=%d(%s) ", uid, pw->pw_name);  /* 사용
      자 ID와 사용자 계정 출력 */
printf("gid=%d(%s) groups=", gid, pg->gr_name);
      /* 그룹 ID와 그룹 이름 출력 */
/* 그룹 파일로부터 모든 그룹 정보를 차례로 얻어 옴 */
   while (pg = getgrent()) {
      /* 해당 그룹에 속한 모든 사용자 계정에 대해 */
      for (i=0; pg->gr_mem[i]!=NULL; i++) {
         /* 프로그램의 사용자 계정(pw->pw_name)과
       pg가 가리키는 그룹에 속한
            사용자 계정이 같다는 것은 현 사용자가 pg
       그룹에 속해있다는 것을
            의미한다. 조건이 참이면 즉, 현 사용자 계정
       이 같으면
            pg가 가리키는 그룹 이름을 출력한다. */
      if (!strcmp(pw->pw_name, pg->gr_mem[i]))
         printf("%d(%s) ", pg->gr_gid, pg->gr_name);
      }
   }
   printf("₩n");
   exit(0);
}
```

- 셸 명령어인 id와 같은 동작