HALLYM UNIVERSITY

Operating Systems LAB 5

Wonpyo Kim

skykwp@gmail.com

Outline

- Substance
  - Command Line Parameter
  - Environment Variable
  - User and Group Information

Command Line Parameter

- 같은 재료가 주어져도 이를 얼마나 효율적으로 활용하느냐에 따라 그 결과는 천차만별이다. 이는 프로그래밍 환경에서도 마찬가지다.

- 리눅스 환경에서 대부분의 쉘 명령어는 옵션과 인수를 선택적으로 사용한다.

- 실행중인 모든 프로그램에 기본적으로 설정된 환경 정보를 알아내고, 이를 자신에게 맞게 새롭게 설정하는 방법에 대해서 알아본다.

- 리눅스 시스템에서는 사용자에 대한 다양한 정보를 관리하고 설정하기 때문에, 사용자에 따라 할 수 있는 일이 정해져 있으므로 이러한 정보를 얻는 것이 중요하다.

Command Line Parameter

- 대부분의 쉘 명령어는 다음과 같이 옵션과 인수를 선택적으로 사용한다.

```
kwp@A1409-OSLAB-01:~/student/lab5$ ls -l
total 4
-rw-rw-r-- 1 kwp kwp 21  3월  28 16:30 test.c
kwp@A1409-OSLAB-01:~/student/lab5$ cat -n test.c
     1
     2  #include <stdio.h>
     3
kwp@A1409-OSLAB-01:~/student/lab5$
```

- C 언어는 기본적으로 명령라인 인수를 받아들일 수 있는 기능을 제공하기 때문에 쉽게 구현할 수 있다.

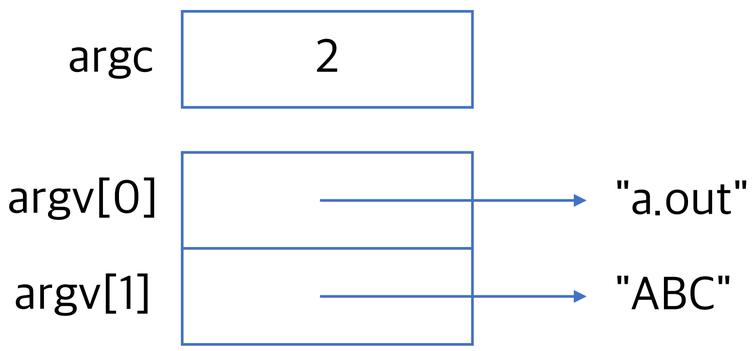
- 다음과 같이 main 함수에 argc 와 argv 를 이용하면 명령라인 인수를 받아들일 수 있다. argc 는 인수의 개수, argv 는 인수의 내용이 배열로 저장된다.

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

}
```

Command Line Parameter

- 다음과 같이 실행하면 argc 는 2가 되고 argv[0] 은 첫 번째 인수인 "a.out" 을 argv[1] 은 두 번째 인수인 "ABC" 를 나타낸다.

```
kwp@A1409-OSLAB-01:~/student/lab5$ a.out ABC
```

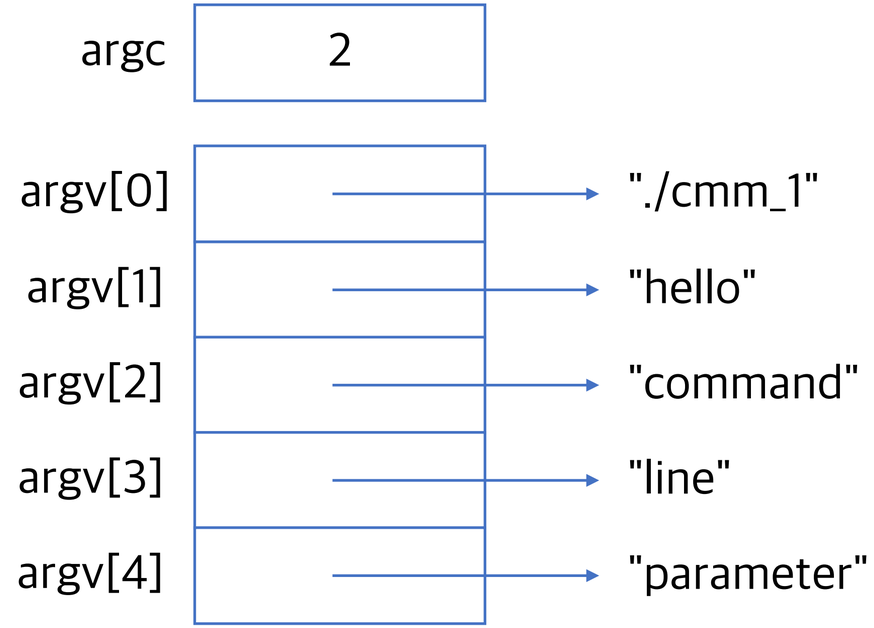


Command Line Parameter

- 다음은 argc, argv를 활용한 프로그램이다.

```c
#include <stdio.h>

int
main(int argc, char *argv[])
{
    int i;

    printf("argc: %d\n", argc);

    for (i = 0; i < argc; i++)
        printf("argv[%d]: %s\n", i, argv[i]);

    return 0;
}
```

```
kwp@A1409-OSLAB-01:~/student/lab5$ ./cmm_1 hello command line parameter
argc: 5
argv[0]: ./cmm_1
argv[1]: hello
argv[2]: command
argv[3]: line
argv[4]: parameter
```



Command Line Parameter

- 실제로 사용하는 프로그램은 명령일때 '-' 로 구분하여 처리한다. 이를 옵션으로 처리하는 방법은 다음과 같다.

```c
#include <stdio.h>

int
main(int argc, char *argv[])
{
    int i;

    for (i = 1; i < argc; i++) {
        if (argv[i][0] == '-') {
            printf("option: %s\n", argv[i]+1);
        } else {
            printf("argument: %s\n", argv[i]);
        }
    }
}
```

```
kwp@A1409-OSLAB-01:~/student/lab5$ ./cmp_3 -l mnist.py "rnn_lstm_gru" -ai
option: l
argument: mnist.py
argument: rnn_lstm_gru
option: ai
```

Command Line Parameter

- 이러한 명령라인 인수를 효율적으로 사용할 수 있는 getopt 함수가 존재한다. 사용법은 조금 복잡하다.

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: 옵션목록
optarg: 인수를 필요로 하는 옵션을 처리할 때 인수를 가리킴
optind: 다음에 처리할 인수의 argv 첨자
opterr: 오류 메시지 출력 여부를 결정하는 변수로 0으로 설정하면 출력되지 않음
optopt: 인식되지 않는 옵션
반환값
성공: 옵션
인식되지 않은 옵션: 물음표(?)
옵션이 없으면: -1
헤더파일
<unistd.h>

Command Line Parameter

- 다음은 여러 옵션들을 처리하는 방법의 예이다.

```c
#include <stdio.h>
#include <unistd.h>

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;
            default:
                printf("unknown option: %c\n", optopt);
        }
    }
}
```

Command Line Parameter

- 실행 예

```
kwp@A1409-OSLAB-01:~/student/lab5$ ./cmp_4 -l -f mnist.py -ai
option: l
option f's argument: mnist.py
option: a
option: i
kwp@A1409-OSLAB-01:~/student/lab5$ ./cmp_4 mnist.py -o
./cmp_4: invalid option -- 'o'
unknown option: o
kwp@A1409-OSLAB-01:~/student/lab5$ ./cmp_4 -f
./cmp_4: option requires an argument -- 'f'
unknown option: f
```

- 별도로 정의하지 않은 옵션이나, 인수가 필요하지만 입력하지 않은 옵션의 경우 getopt 함수의 전역변수로 정보를 출력할 수 있다.

Command Line Parameter

- 앞서 구현한 명령라인 인수를 다루는 방법을 활용하여 cat 과 같은 동작을 하는 프로그램을 작성할 수 있다.

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

int flag = 0;

static void output_file(FILE *fp);

void
main(int argc, char *argv[])
{
    FILE *fp;
    int opt;

    if (argc == 1) {
        printf("usage: ./cmp_cat [-n] filename ...\n");
        exit(1);
    }

    while ((opt = getopt(argc, argv, "n")) != -1) {
        switch (opt) {
            case 'n':
                flag = 1;
                break;
            default:
                printf("unknown option: %c\n", optopt);
        }
    }

    if (flag == 1) {
        argv++;
    }
```

```c
    while (*++argv) {
        printf("\n[filename: %s]\n\n", *argv);
        if ((fp = fopen(*argv, "r")) == NULL) {
            perror("fopen failed");
            exit(2);
        }
        output_file(fp);
        fclose(fp);
    }

    exit(0);
}

static void
output_file(FILE *fp)
{
    int ch, line = 1;

    if (flag) {
        printf("1  ");
    }
    while ((ch = getc(fp)) != EOF) {
        if (flag && ch == '\n') {
            if ((ch = getc(fp)) == EOF) {
                break;
            }
            ungetc(ch, fp);
            printf("\n%-4d", ++line);
        } else {
            putc(ch, stdout);
        }
    }
    printf("\n");
}
```

Command Line Parameter

- 실행 예

```
kwp@A1409-OSLAB-01:~/student/lab5$ ./cmp_cat -n test.c

[filename: test.c]

1
2   #include <stdio.h>
3
4   int main(int argc, char **argv)
5   {
6
7   }
8
9
```

Outline

- Substance
  - Command Line Parameter
  - Environment Variable
  - User and Group Information

Environment Variable

- 실행중인 모든 프로그램에는 기본적으로 설정된 정보가 있는데, 이를 저장한 변수를 환경변수라 한다. 이를 확인하는 쉘 명령어는 printenv 로 다음과 같다.

```
kwp@A1409-OSLAB-01:~$ printenv
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:
ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;
31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01
;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.t
z=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;3
1:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mj
pg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=0
1;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;3
5:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.a
sf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35
:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.mid
i=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;
36:
LC_MEASUREMENT=ko_KR.UTF-8
SSH_CONNECTION=210.115.226.231 50581 210.115.226.230 5150
LESSCLOSE=/usr/bin/lesspipe %s %s
LC_PAPER=ko_KR.UTF-8
LC_MONETARY=ko_KR.UTF-8
LANG=en_US.UTF-8
LC_NAME=ko_KR.UTF-8
```

- 환경변수는 "PWD=/home/kwp/test/env" 와 같은 형태로 설정되어 있다.

Environment Variable

- 앞 장에서 명령라인의 argc, argv에 대하여 알아보았다. envp 라는 인수도 있기 때문에 완전한 main 함수는 다음과 같은 형태를 갖는다.

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

}
```

- envp 인수는 현재 설정되어 있는 환경변수를 모두 갖고있다. 아래는 이를 출력하는 프로그램의 예이다.

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

main(int argc, char *argv[], char *envp[])
{
    while (*envp) {
        printf("%s\n", *envp++);
    }
}
```

Environment Variable

- 시스템에서는 envp와 같은 의미를 갖는 변수인 environ도 존재한다.

environ 변수
기능
　　환경 변수 정보를 저장한다.
기본형
　　extern char **environ;
헤더파일
　　<unistd.h>

- 이를 호출하여 출력하는 프로그램은 다음과 같다.

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

extern char **environ;

main()
{
    while (*environ) {
        printf("%s\n", *environ++);
    }
}
```

Environment Variable

- 환경 변수와 관련된 함수는 getenv, putenv, setenv, unsetenv 가 있다.

getenv 함수
기능
　　환경 변수값을 가져온다.
기본형
　　char *getenv(const char *name);
　　name: 알고자하는 환경 변수 이름
반환값
　　성공: 환경 변수값
　　실패: NULL
헤더파일
　　<stdlib.h>

Environment Variable

- 다음은 환경변수 HOME, PWD, LINUX에 대한 값을 가져온다.

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

void
main(void)
{
    char *home_dir, *work_dir, *tmp;

    if ((home_dir = getenv("HOME")) != NULL) {
        printf("home directory: %s\n", home_dir);
    }
    if ((work_dir = getenv("PWD")) != NULL) {
        printf("working directory: %s\n", work_dir);
    }
    if ((tmp = getenv("LINUX")) != NULL) {
        printf("temp: %s\n", tmp);
    }
}
```

```
kwp@A1409-OSLAB-01:~/student/lab5$ ./homepwdlinux
home directory: /home/kwp
working directory: /home/kwp/student/lab5
```

- LINUX 라는 환경 변수는 존재하지 않으므로 출력되는 것이 없다.

Environment Variable

- putenv 함수는 환경 변수의 값을 변경하는 함수로, string 은 "변수이름=변수값" 형식이어야 한다. putenv 함수로 설정된 환경 변수값은 프로그램이 종료되면 해제되고 실행하기 전의 상태로 돌아간다.

putenv 함수
기능
	환경 변수값을 변경한다.
기본형
	int putenv(const char *string);
	string: 변경하고자 하는 환경 변수 정보로 "변수이름=변수값" 형식
반환값
	성공: 0
	실패: -1
헤더파일
	<stdlib.h>

Environment Variable

- 실행 예

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

main()
{
    char *home_dir;

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

    putenv("HOME=/home/kwp/student");

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

```
kwp@A1409-OSLAB-01:~/student/lab5$ ./putenv
home directory: /home/kwp
home directory: /home/kwp/student
```

Environment Variable

- setenv 함수는 변수 이름과 변수값을 분리해서 전달한다. 즉, 이미 해당 변수가 존재할 경우, overwrite 값이 0이면 값을 변경하지 않고 1이면 변경한다.

setenv 함수
기능
	환경 변수값을 변경한다.
기본형
	int setenv(const char *name, const char *value, int overwrite);
	name: 변경하고자 하는 환경 변수 이름
	value: 새롭게 설정할 환경 변수값
	overwrite: 변수가 이미 존재할 때, 변경할지 여부를 결정하는 인수로 0이면 변경하지 않음
반환값
	성공: 0
	실패: -1
헤더파일
	<stdlib.h>

Environment Variable

- 실행 예

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

main()
{
    char *value;

    setenv("HOME", "/home/kwp/student", 0);
    value = getenv("HOME");
    printf("[setenv overwrite: 0] HOME: %s\n", value);

    setenv("HOME", "/home/kwp/student", 1);
    value = getenv("HOME");
    printf("[setenv overwrite: 1] HOME: %s\n", value);

    exit(0);
}
```

```
kwp@A1409-OSLAB-01:~/student/lab5$ ./setenv
[setenv overwrite: 0] HOME: /home/kwp
[setenv overwrite: 1] HOME: /home/kwp/student
```

Environment Variable

- unsetenv 함수는 환경변수 자체를 삭제하는 함수이다.

```
unsetenv 함수
기능
        환경변수 값을 삭제한다.
기본형
        void unsetenv(const char *name);
        name: 삭제하고자 하는 환경변수 이름
반환값
        없음
헤더파일
        <stdlib.h>
```

Environment Variable

- 실행 예

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

main()
{
    char *value;

    if ((value = getenv("HOME")) != NULL) {
        printf("HOME: %s\n", value);
    } else {
        printf("HOME: no value\n");
    }

    unsetenv("HOME");

    if ((value = getenv("HOME")) != NULL) {
        printf("HOME: %s\n", value);
    } else {
        printf("HOME: no value\n");
    }

    exit(0);
}
```

```
kwp@A1409-OSLAB-01:~/student/lab5$ ./unsetenv
HOME: /home/kwp
HOME: no value
```

Outline

- Substance
  - **Command Line Parameter**
  - **Environment Variable**
  - **User and Group Information**

User and Group Information

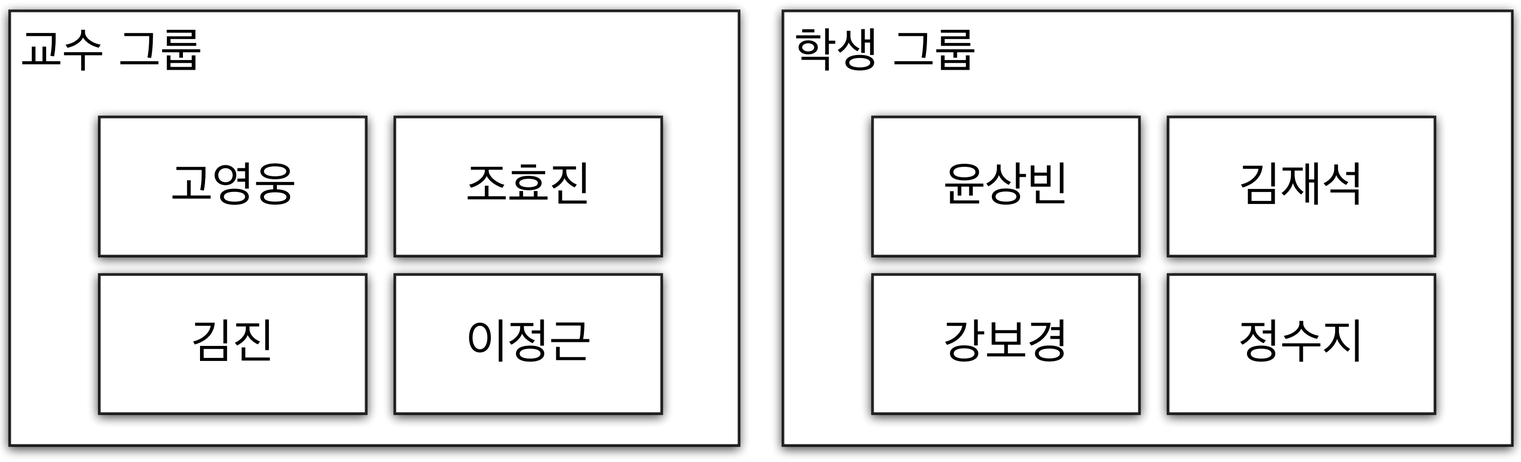
- 리눅스 시스템을 사용하기 위해서 사용자 계정을 만들 때, 각 계정마다 번호가 부여되는데, 이를 사용자 ID(User ID)라고 한다. 실행중에 있는 프로그램의 사용자 ID를 반환하는 함수로는 getuid 와 geteuid 가 있다.

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
헤더파일
　　<unistd.h>
　　<sys/types.h>

User and Group Information

- 가령, 팀 단위로 프로젝트를 개발할 때 팀원들은 파일을 공유하고 그 외 사용자들은 파일을 이용하지 못하게 하길 원하는 경우가 있다. 이 때, 그룹을 이용하면 된다.

- 리눅스에서는 여러 사용자를 모아 그룹으로 관리하는 기능을 제공하고 있다.



- 사용자의 그룹할당은 슈퍼유저가 담당한다. 이러한 그룹에도 유일한 번호가 부여된다. 이를 그룹 ID(Group ID)라 한다.

User and Group Information

- 실행 중에 있는 프로그램의 그룹 ID를 반환하는 함수는 getgid, getegid 이다.

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
헤더파일
	<unistd.h>
	<sys/types.h>

User and Group Information

- 실행 예, getuid, getgid 함수를 이용하여 실제 사용자 ID와 실제 그룹 ID를 출력한다.

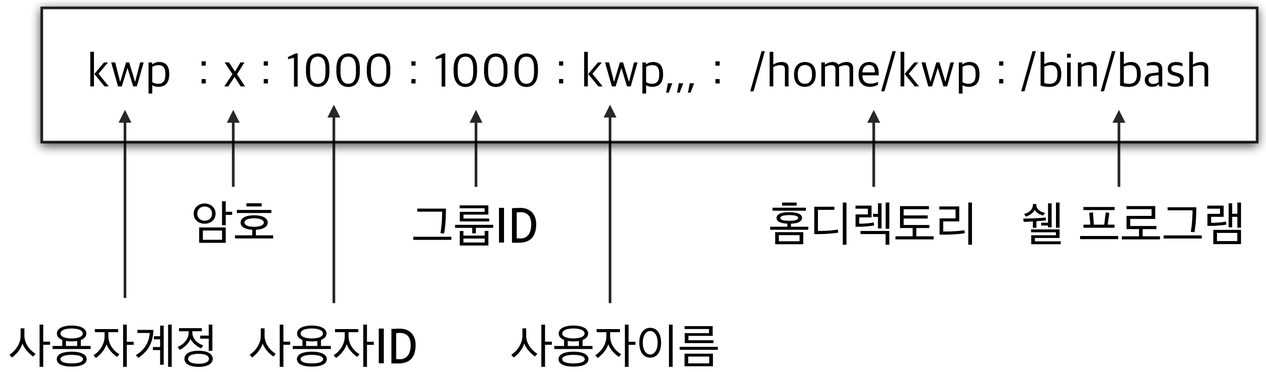
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

main()
{
    printf("Useid: %d\n", getuid());
    printf("Groupid: %d\n", getgid());
    exit(0);
}
```

```
kwp@A1409-OSLAB-01:~/student/lab5$ ./guidgid
Useid: 1000
Groupid: 1000
```

User and Group Information

- 사용자들에 대한 정보는 /etc/passwd 파일(패스워드 파일)에 있다. 각 사용자들에 대한 정보는 행 단위로 있으며 콜론(:)으로 항목을 구분한다. 다음은 패스워드 파일의 예이다.

```
kwp:x:1000:1000:kwp,,,:/home/kwp:/bin/bash
sshd:x:122:65534::/run/sshd:/usr/sbin/nologin
s_20151539:x:1001:1001::/home/s_20151539:/bin/bash
s_20142108:x:1002:1002::/home/s_20142108:/bin/bash
s_20165163:x:1003:1003::/home/s_20165163:/bin/bash
s_20135151:x:1004:1004::/home/s_20135151:/bin/bash
s_20135146:x:1005:1005::/home/s_20135146:/bin/bash
s_20135148:x:1006:1006::/home/s_20135148:/bin/bash
s_20145347:x:1007:1007::/home/s_20145347:/bin/bash
s_20155204:x:1008:1008::/home/s_20155204:/bin/bash
s_20155205:x:1009:1009::/home/s_20155205:/bin/bash
s_20181214:x:1010:1010::/home/s_20181214:/bin/bash
s_20175103:x:1011:1011::/home/s_20175103:/bin/bash
s_20175301:x:1012:1012::/home/s_20175301:/bin/bash
s_20145103:x:1013:1013::/home/s_20145103:/bin/bash
s_20155105:x:1014:1014::/home/s_20155105:/bin/bash
s_20145182:x:1015:1015::/home/s_20145182:/bin/bash
s_20175120:x:1016:1016::/home/s_20175120:/bin/bash
s_20175121:x:1017:1017::/home/s_20175121:/bin/bash
s_20175126:x:1018:1018::/home/s_20175126:/bin/bash
s_20145128:x:1019:1019::/home/s_20145128:/bin/bash
s_20155127:x:1020:1020::/home/s_20155127:/bin/bash
s_20155131:x:1021:1021::/home/s_20155131:/bin/bash
s_20145184:x:1022:1022::/home/s_20145184:/bin/bash
s_20155138:x:1023:1023::/home/s_20155138:/bin/bash
s_20155329:x:1024:1024::/home/s_20155329:/bin/bash
s_20145156:x:1025:1025::/home/s_20145156:/bin/bash
s_20135173:x:1026:1026::/home/s_20135173:/bin/bash
s_20135126:x:1027:1027::/home/s_20135126:/bin/bash
s_20155157:x:1028:1028::/home/s_20155157:/bin/bash
```



User and Group Information

- 각 사용자에 대한 정보는 struct passwd 데이터형으로 표현되며, <pwd.h> 헤더파일 앞에 정의되어 있다.

```
/* The passwd structure.  */
struct passwd
{
  char *pw_name;                /* Username.  */
  char *pw_passwd;              /* 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.  */
};
```

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
헤더파일
	<pwd.h>
	<sys/types.h>

User and Group Information

- getpwuid 함수를 이용해 계정, 사용자 ID, 그룹 ID, 홈 디렉토리를 출력한다.

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

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

    uid = getuid();
    pw = getpwuid(uid);
    printf("name:%s, uid:%d, gid:%d, home:%s\n", \
        pw->pw_name, pw->pw_uid, pw->pw_gid, pw->pw_dir);

    exit(0);
}
```

```
kwp@A1409-OSLAB-01:~/student/lab5$ ./getpw
name:kwp, uid:1000, gid:1000, home:/home/kwp
```

User and Group Information

- 앞선 예제를 getpwnam 함수를 이용해서 구현할 때는 사용자 계정이 필요하다. 이 때, getlogin 함수를 이용한다.

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

- 적용은 단순이 아래 라인을 getpwnam 함수로 바꾸면된다.

```
pw = getpwuid(uid);
```

User and Group Information

- 패스워드 파일을 이용하는 또 다른 함수에는 getpwent, setpwent, endpwent 가 있다.

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

- 즉, getpwent 함수는 /etc/passwd 의 내용을 한 번 호출할 때 마다 한 줄씩 가져온다.

User and Group Information

- setpwent 함수는 패스워드 파일을 열고 시작 지점으로 돌아가게 하는 함수로 패스워드 파일로부터 사용자 정보를 얻다가 첫 사용자로 돌아가고자 할 때 사용한다.

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

User and Group Information

- 패스워드 파일에 대한 사용이 끝나면 닫아주는 것이 바람직하다. 이 때, 사용하는 함수는 endpwent 이다.

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

User and Group Information

- 실행 예

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

main()
{
    struct passwd *pw;

    while (pw = getpwent()) {
        printf("name:%s, uid:%d, home:%s\n", \
            pw->pw_name, pw->pw_uid, pw->pw_dir);
    }

    endpwent();
    exit(0);
}
```

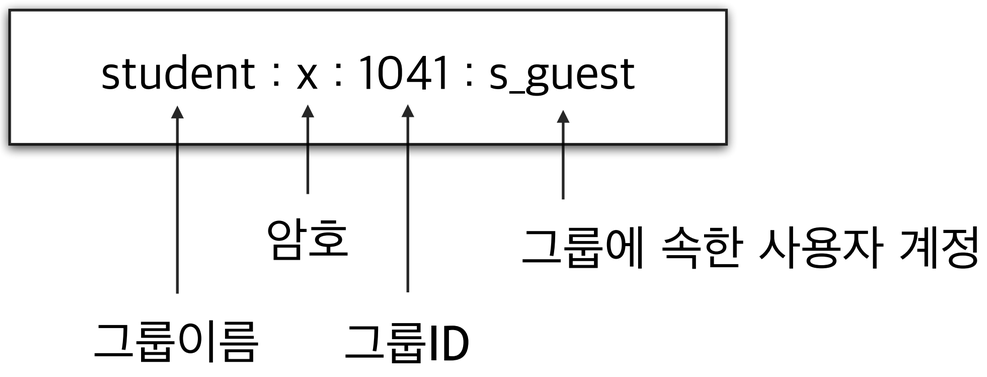
```
kwp@A1409-OSLAB-01:~/student/lab5$ ./pwent
name:root, uid:0, home:/root
name:daemon, uid:1, home:/usr/sbin
name:bin, uid:2, home:/bin
name:sys, uid:3, home:/dev
name:sync, uid:4, home:/bin
name:games, uid:5, home:/usr/games
name:man, uid:6, home:/var/cache/man
name:lp, uid:7, home:/var/spool/lpd
name:mail, uid:8, home:/var/mail
name:news, uid:9, home:/var/spool/news
name:uucp, uid:10, home:/var/spool/uucp
name:proxy, uid:13, home:/bin
name:www-data, uid:33, home:/var/www
name:backup, uid:34, home:/var/backups
```

User and Group Information

- 그룹에 대한 정보는 /etc/group 파일(그룹 파일)에 있다. 각각의 그룹에 대한 정보는 행 단위로 있고 콜론(:)으로 항목을 구분한다.

```
s_20185138:x:1036:
s_20175147:x:1037:
s_20155260:x:1038:
s_20145113:x:1039:
s_guest:x:1040:
student:x:1041:s_guest,s_20151539,s_20142108,s_20165163,s_20135151,s_20135146,s_20135148,s_20145347,s_20155204,s_2015
5205,s_20181214,s_20175103,s_20175301,s_20145103,s_20155105,s_20145182,s_20175120,s_20175121,s_20175126,s_20145128,s_
20155127,s_20155131,s_20145184,s_20155138,s_20155329,s_20145156,s_20135173,s_20135126,s_20155157,s_20165526,s_2016516
8,s_20135303,s_20155321,s_20155330,s_20135147,s_20185242,s_20185138,s_20175147,s_20155260,s_20145113
```

- 각 행은 그룹이름, 그룹암호, 그룹ID, 이 그룹에 속한 사용자 계정의 4개의 항목으로 이루어져 있다.



User and Group Information

- 각 그룹에 대한 정보는 다음과 같이 struct group 데이터형으로 표현되며, <grp.h> 헤더파일에 정의되어있다.

```
/* The group structure.  */
struct group
  {
    char *gr_name;              /* Group name.  */
    char *gr_passwd;            /* Password.    */
    __gid_t gr_gid;             /* Group ID.    */
    char **gr_mem;              /* Member list. */
};
```

User and Group Information

- 그룹 파일로부터 그룹에 대한 정보를 처음부터 하나씩 얻어오는 함수는 getgrent 가 있다. 호출할 때 마다 정보를 얻어온다.

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

User and Group Information

- setgrent 는 그룹파일의 시작지점으로 돌아가는 함수이며, entgrent 는 열었던 그룹파일을 닫는 함수이다.

```
setgrent, endgrent 함수
기능
        setgrent는 그룹파일의 시작 지점으로 돌아가게 되고, endgrent는 그룹파일을 닫는다.
기본형
        void setgrent(void);
        void endgrent(void);
반환값
        없음
헤더파일
        <grp.h>
        <sys/types.h>
```

User and Group Information

- 실행 예

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

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

    while (pg = getgrent()) {
        printf("group name: %s\n  member: ", pg->gr_name);
        for (i = 0; pg->gr_mem[i] != NULL; i++) {
            printf("%s ", pg->gr_mem[i]);
        }
        printf("\n\n");
    }
    endgrent();
    exit(0);
}
```

```
group name: student
  member: s_guest s_20151539 s_20142108 s_20165163 s_20135151 s_20135146 s_20135148 s_20145347 s_20155204 s_20155205
s_20181214 s_20175103 s_20175301 s_20145103 s_20155105 s_20145182 s_20175120 s_20175121 s_20175126 s_20145128 s_20155
127 s_20155131 s_20145184 s_20155138 s_20155329 s_20145156 s_20135173 s_20135126 s_20155157 s_20165526 s_20165168 s_2
0135303 s_20155321 s_20155330 s_20135147 s_20185242 s_20185138 s_20175147 s_20155260 s_20145113

kwp@A1409-OSLAB-01:~/student/lab5$
```

User and Group Information

- 그룹과 관련된 다른 함수에는 getgrgid, getgrnam 이 있으며, 해당 그룹에 대한 정보를 읽어온다. getgrgid는 그룹 ID를 이용하고 getgrnam은 그룹 계정을 이용한다.

```
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 얻고자하는 그룹 ID
        name: 정보를 얻고자하는 그룹 이름
반환값
        성공: 그룹 정보에 대한 포인터
        실패: NULL
헤더파일
        <grp.h>
        <sys/types.h>
```

User and Group Information

- 사용자와 그룹 정보를 알아내는 프로그램의 예

```c
#include <stdio.h>
#include <stdlib.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();
    pw = getpwuid(uid);
    gid = getgid();
    pg = getgrgid(gid);

    printf("uid=%d(%s) ", uid, pw->pw_name);
    printf("gid=%d(%s) groups=", gid, pg->gr_name);

    while (pg = getgrent()) {
        for (i = 0; pg->gr_mem[i] != NULL; i++) {
            if (!strcmp(pw->pw_name, pg->gr_mem[i])) {
                printf("%d(%s) ", pg->gr_gid, pg->gr_name);
            }
        }
    }

    printf("\n");
    exit(0);
}
```

```
kwp@A1409-OSLAB-01:~/student/lab5$ ./pwpg
uid=1000(kwp) gid=1000(kwp) groups=4(adm) 24(cdrom) 27(sudo) 30(dip) 46(plugdev) 116(lpadmin) 126(sambashare)
```