리눅스 사용

기본적인 사용

간단한 명령 실행

- 우분투 (Ubuntu) 리눅스
- 우분투 리눅스의 데스크탑
- 터미널 프로그램의 사용
  - 명령줄 (command line)
- 터미널에서의 기본적인 명령 실행

```
$ date
$ hostname
$ uname
$ who
$ ls
$ clear
```

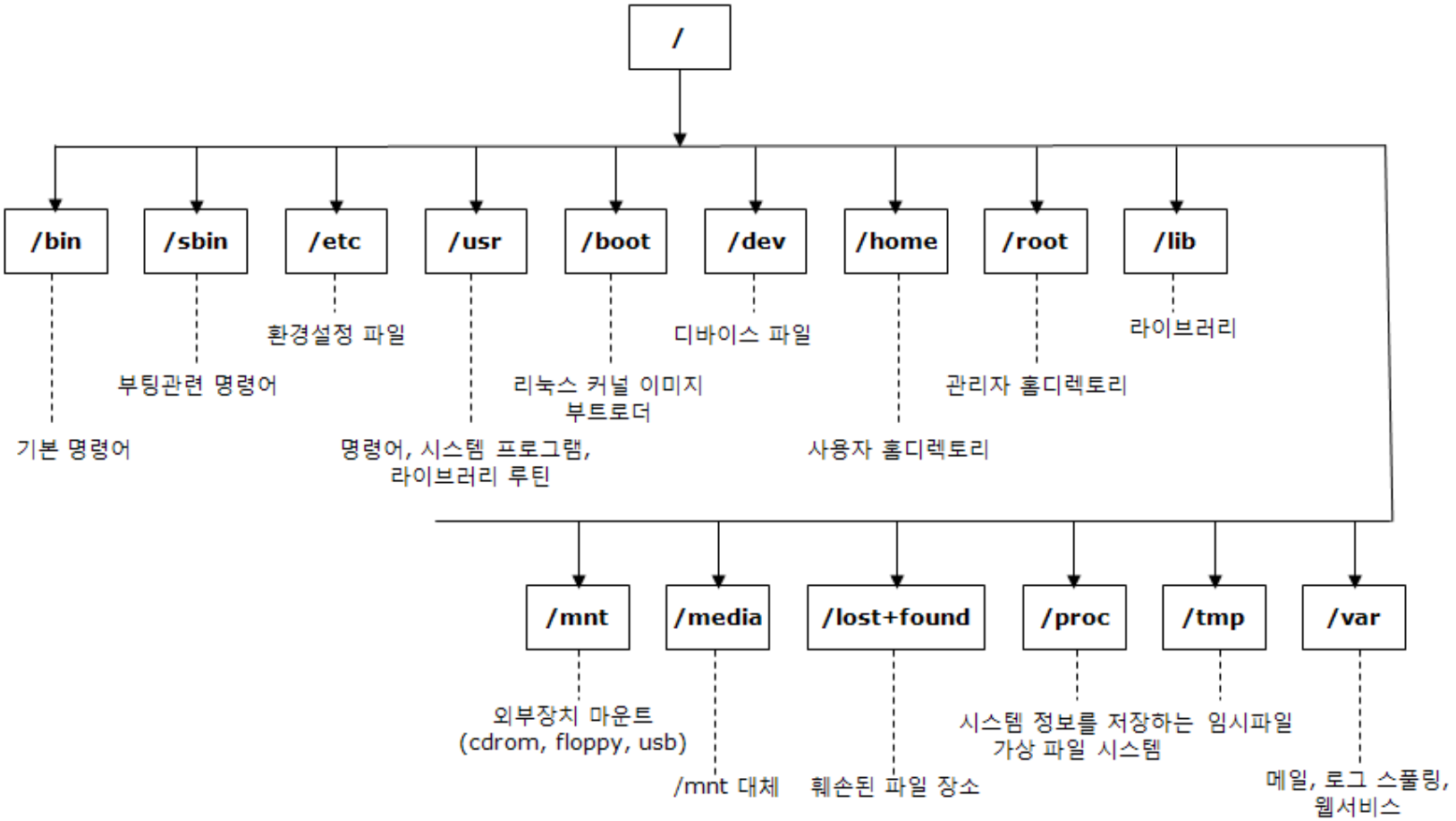
파일 및 디렉터리의 개념

파일의 종류

- 일반 파일(regular file)
  - 데이터를 가지고 있으면서 디스크에 저장된다.
- 디렉터리(directory)
  - 윈도우의 폴더에 해당
  - 디렉터리 자체도 하나의 파일로 한 디렉터리는 다른 디렉터리들을 포함함으로써 계층 구조를 이룬다.
  - 부모 디렉터리는 다른 디렉터리들을 서브 디렉터리로 갖는다.
- 특수 파일(special file)
  - 물리적인 장치에 대한 내부적인 표현
  - 키보드(stdin), 모니터(stdout), 프린터 등도 파일처럼 사용

디렉터리 계층구조

- 리눅스의 디렉터리는 루트로부터 시작하여 계층구조를 이룬다.



홈 디렉터리/현재 작업 디렉터리

- 홈 디렉터리(home directory)
  - 사용자 계정마다 주어지는 기본 디렉터리 – 각 사용자마다 별도의 홈 디렉터리가 있음
  - 사용자가 로그인하면 홈 디렉터리에서 작업을 시작함

- 현재 작업 디렉터리(current working directory)
  - 현재 작업 중인 디렉터리
  - 로그인 직후에는 홈 디렉터리가 현재 작업디렉터리이다.

디렉터리 관련 명령

- pwd(print working directory)
  - 현재 작업 디렉터리를 프린트

```
$ pwd
```

- cd(change directory)
  - 현재 작업 디렉터리를 이동

```
$ cd [디렉터리]
```

- mkdir(make directory)
  - 새 디렉터리를 만듬

```
$ mkdir 디렉터리
```

디렉터리 리스트

- ls(list)
  - 디렉터리의 내용을 리스트
- $ ls

  cs1.txt
- $ ls -s -s(size)

  총 6

  6 cs1.txt
- $ ls -a -a(all)

  . .. cs1.txt

디렉터리 리스트

- $ ls -l -l(long)

```
-rw-r--r-- 1 chang faculty 2088 4월 16일 13:37 cs1.txt
```

- $ ls -asl

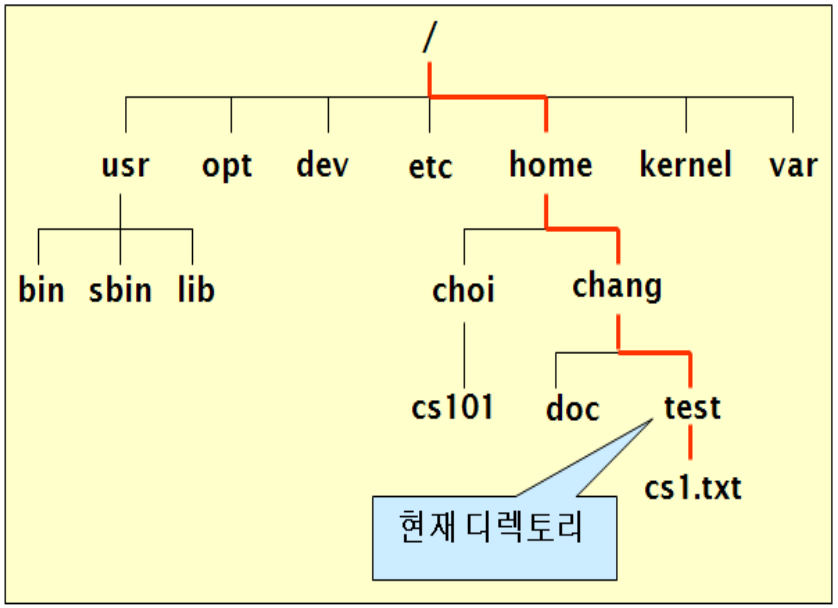
```
총 10
2 drwxr-xr-x 2 chang faculty 512 4월 16일 13:37 .
2 drwxr-xr-x 3 chang faculty 512 4월 16일 13:37 ..
6 -rw-r--r-- 1 chang faculty 2088 4월 16일 13:37 cs1.txt
```

디렉터리 관련 명령어

명령어	의미
ls	파일 및 디렉터리 리스트
ls -a	모든 파일과 디렉터리 리스트
ls -asl	모든 파일 자세히 리스트
mkdir	디렉터리 만들기
cd 디렉터리	디렉터리로 이동
cd	홈 디렉터리로 이동
cd ~	홈 디렉터리로 이동
cd ..	부모 디렉터리로 이동
pwd	현재 작업 디렉터리 프린트

경로명

- 파일이나 디렉터리에 대한 정확한 이름
- 절대 경로명(absolute pathname)
  - 루트 디렉터리로부터 시작하여 경로 이름을 정확하게 적는 것
- 상대 경로명(relative path name)
  - 현재 작업 디렉터리부터 시작해서 경로 이름을 적는 것



~ : 홈 디렉터리
. : 현재 디렉터리
.. : 부모 드렉터리

cs1.txt의 절대 경로명
/home/chang/test/cs1.txt

cs1.txt의 상대 경로명
cs1.txt

파일 내용 리스트

- 파일 내용 출력과 관련된 다음 명령어들
  - cat, more, head, tail, wc, 등

```
$ 명령어 파일

$ 명령어 파일*

$ more 파일+
```

cat 명령어

- 파일 내용 출력

```
$ cat cs1.txt

$ cat
...
^D

$ cat > cs1.txt
...
^D
```

more/head/tail/wc

- more 명령어
  하나 이상의 파일 이름을 받을 수 있으며 각 파일의 내용을 페이지 단위로 출력
- head 명령어
  파일의 앞부분(10줄)을 출력한다.
- tail 명령어
  파일의 뒷부분(10줄)을 출력한다.

- wc(word count)
  파일에 저장된 줄, 단어, 문자의 개수를 세서 출력

```
$ wc cs1.txt
38 318 2088 cs1.txt
```

cp 명령어

- $ cp 파일1 파일2
  파일1의 복사본 파일2를 현재 디렉터리 내에 만듬

```
$ cp cs1.txt cs2.txt
$ ls -l cs1.txt cs2.txt
-rw-r--r-- 1 chang faculty 2088 4월 16일 13:37 cs1.txt
-rw-r--r-- 1 chang faculty 2088 4월 16일 13:45 cs2.txt
```

- $ cp 파일 디렉터리
  파일1의 복사본을 디렉터리 내에 만듬

```
$ cp cs1.txt /tmp
```

mv 명령어

- mv(move)
  파일1의 이름을 파일2로 변경한다.

```
$ mv 파일1 파일2
$ mv cs2.txt cs3.txt
$ ls -l
-rw-r--r-- 1 chang faculty 2088 4월 16일 13:37 cs1.txt
-rw-r--r-- 1 chang faculty 2088 4월 16일 13:56 cs3.txt
```

- 파일을 디렉터리 내로 이동

```
$ mv 파일 디렉터리
$ mv cs3.txt /tmp
```

파일/디렉터리 삭제

- rm(remove) 명령어
  명령줄 인수로 받은 파일(들)을 지운다.

  $ rm 파일+
  $ rm cs1.txt

- $ rm -r 디렉터리
  디렉터리 내의 모든 파일 및 하위 디렉터리들을 단번에 지운다.

- rmdir(remove directory) 명령어
  명령줄 인수로 받은 디렉터리(들)을 지운다.
  $ rmdir 디렉터리+
  주의: 디렉터리 내에 아무 것도 없어야 한다.
  $ rmdir test

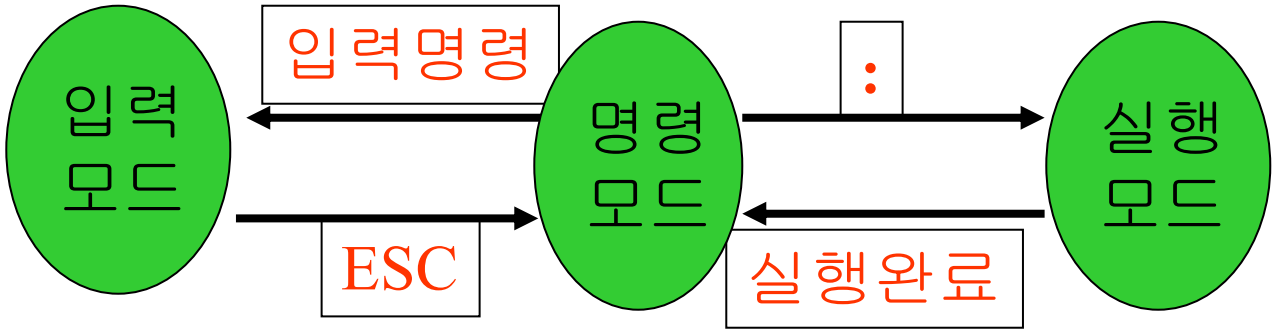
파일 관련 명령어

명령어	의미
cat 파일*	파일 디스플레이
more 파일+	한 번에 한 페이지씩 디스플레이
head 파일*	파일의 앞부분 디스플레이
tail 파일*	파일의 뒷부분 디스플레이
wc 파일*	줄/단어/문자 수 세기
cp 파일1 파일2	파일1을 파일2로 복사
mv 파일1 파일2	파일1을 파일2로 이름 변경
rm 파일+	파일 삭제
rmdir 디렉터리+	디렉터리 삭제
grep 키워드 파일	파일에서 키워드 찾기

vi 문서편집기

vi 편집기

- 실행 방법: vi 명령어 뒤에 파일 이름
- vi 실행 시 시작하는 모드는 명령 모드
- 편집기 모드
  - 입력모드 혹은 편집모드 - 글자를 입력할 수 있는 모드
    - 모드변환방법 - 명령모드에서 a,A,i,o,O를 입력 했을 때
  - 명령모드 혹은 ESC모드 - 커서이동 및 기타 명령어처리
    - 모드변환방법 - 실행모드 혹은 입력모드에서 ESC키를 눌렀을 때
  - 실행모드 혹은 콜론모드 - 내용바꾸기 및 기타
    - 모드변환방법 - 명령모드에서 콜론(:)을 입력했을 때



vi - 명령 모드

- 입력 모드 전환

a : 커서 위치의 다음 칸부터 입력하기(**append**)
A : 커서가 있는 줄의 끝부터 입력하기
i : 커서 위치부터 입력하기 (키보드의 **Insert**도 같은 기능을 합니다.)
I : 커서가 있는 줄의 맨 앞에서부터 입력하기
o : 커서 바로 아래에 줄을 만들고 입력하기(**open line**)
O : 커서 바로 위에 줄을 만들고 입력하기
s : 커서가 있는 단어를 지우고 입력하기
S : 커서가 있는 행을 지우고 입력하기

- 커서 이동

h : 왼쪽, **j** : 위로, **k** : 아래로, **l** : 오른쪽 (방향키 사용 가능) - 글자단위
w : 다음단어로, **b** : 이전단어로 - **word** 단위
^ : 맨 왼쪽의 첫 글자, **$** : 마지막글자의 끝 - 행 단위
^F: 한화면 아래로, **^B**:한 화면 위로, **^D**: 반 화면 아래로, **^U**: 반화면 위로

vi - 명령 모드

- 삭제 기능

x : 커서 위치의 글자 삭제
X : 커서 바로 앞의 글자 삭제
dw : 한 단어를 삭제
D : d$ 커서 위치부터 줄의 끝까지 삭제
dd : 커서가 있는 줄을 삭제

- 복사 및 붙여넣기

yw : 커서 위치부터 단어의 끝까지 복사하기
y0 : 커서 위치부터 줄의 처음까지 복사하기
y$: 커서 위치부터 줄의 끝까지 복사하기
yy : 커서가 있는 줄을 복사하기
yj : 커서가 있는 줄과 그 다음 줄을 복사하기
yk : 커서가 있는 줄과 그 앞줄을 복사하기
p : 커서의 다음 위치에 붙여 넣기
P : 커서가 있는 위치에 붙여 넣기

vi - 명령 모드

- 기타

u : 작업 취소하기 **(undo)**
U : 그 줄에 행해진 작업 모두 취소하기
. : 조금 전에 했던 명령을 반복하기
~ : 대소문자 전환
/검색어 : 아래 방향으로 찾기 **(검색)**
?검색어 : 위쪽 방향으로 찾기
n : 다음 찾기

vi - 실행 모드

- 치환관련 실행

:s/old/new/g - old를 new 로 치환
:s/^old/new/g - 행의 첫 단어가 old 인 것을 new 로 치환
:s/old$/new/g - 행의 끝 단어가 old 인 것을 new 로 치환
:s/aaa//g - aaa를 삭제

- 파일 관련 실행

:w 파일명	**"파일명"으로 저장**
:q	**저장하지 않고 종료**
:q!	**변경 사항을 버리고 종료**
:e 파일명	**"파일명"의 파일을 불러들여 편집**
:r 파일명	**"파일명"의 파일을 읽어서 삽입**
:!명령어	**외부명령어 실행**

파일 속성

파일 속성(file attribute)

- 파일의 이름, 타입, 크기, 소유자, 사용권한, 수정 시간

```
$ ls -sl cs1.txt
6 -rw-r--r-- 1 chang faculty 2088 4월 16일 13:37 cs1.txt
```

파일 속성	의미
블록 수	파일의 블록 수
파일 타입	일반 파일(-), 디렉터리(d), 링크(l), 파이프(p), 소켓(s), 디바이스(b 혹은 c) 등의 파일 종류를 나타낸다.
사용권한	소유자, 그룹, 기타 사용자의 파일에 대한 읽기/쓰기/실행 권한
소유자 및 그룹	파일의 소유자 및 소유자가 속한 그룹
크기	파일을 구성하는 블록 수
수정 시간	파일을 최후로 생성 혹은 수정한 시간

사용권한(permission mode)

- 읽기(r), 쓰기(w), 실행(x) 권한

권한	파일	디렉터리
r	파일에 대한 읽기 권한	디렉터리 내에 있는 파일명을 읽을 수 있는 권한
w	파일에 대한 쓰기 권한	디렉터리 내에 파일을 생성하거나 삭제할 수 있는 권한
x	파일에 대한 실행 권한	디렉터리 내로 탐색을 위해 이동할 수 있는 권한

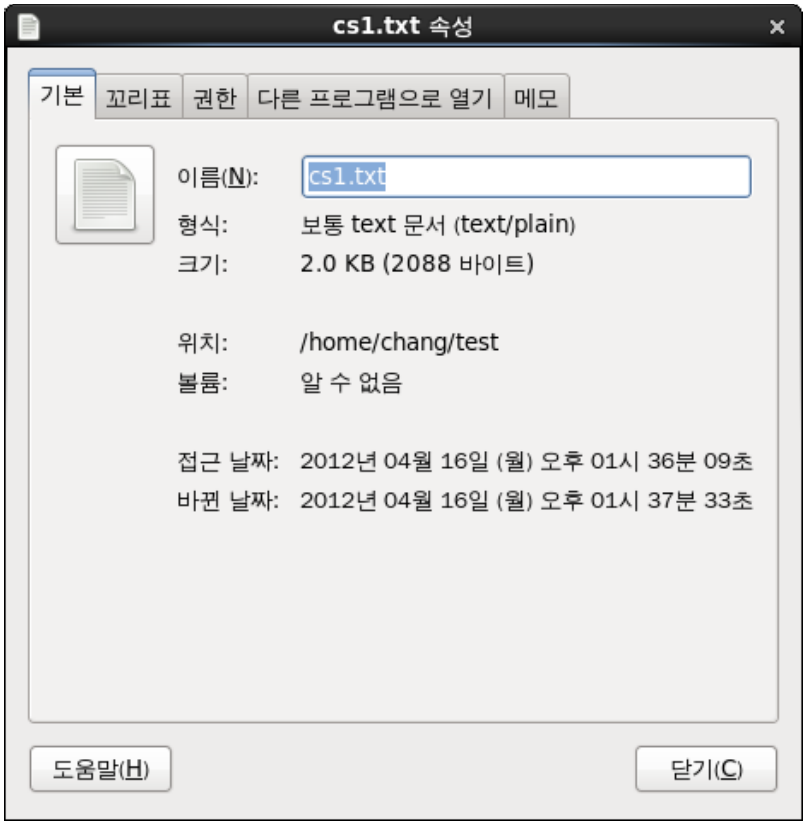
- 파일의 사용권한은 소유자(owner)/그룹(group)/기타(others)로 구분하여 관리한다.

- 예

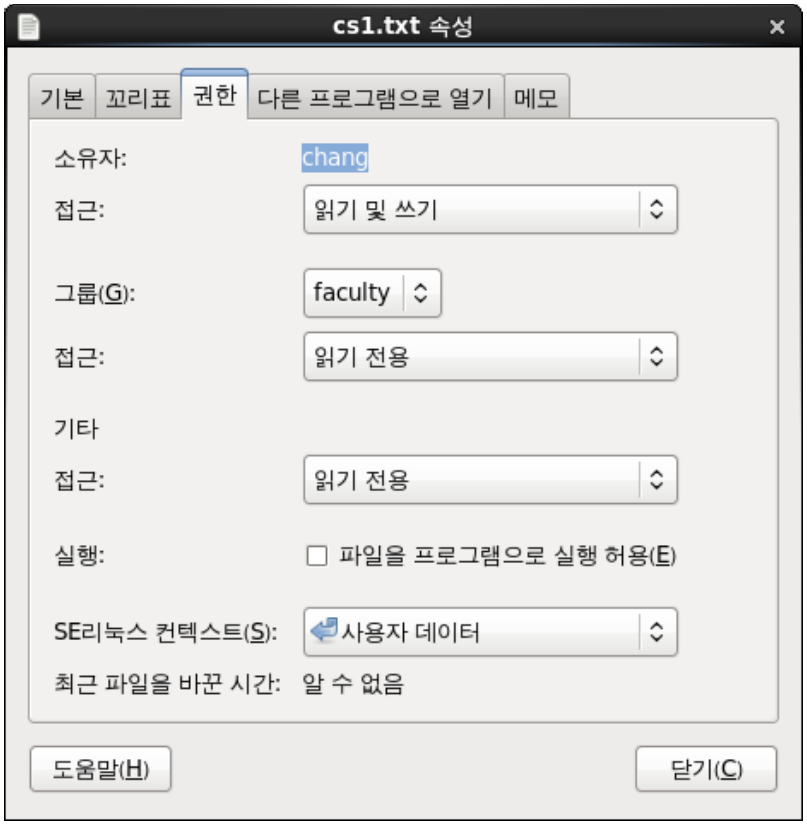
```
소유자 그룹 기타
rw-    r--  r--
```

X 윈도우의 GNOME 데스크톱에서 속성 확인

기본 속성



사용권한



chmod(change mode)

- 파일 혹은 디렉터리의 사용권한을 변경하는 명령어

  $ chmod [-R] 사용권한 파일

  -R 옵션은 디렉터리 내의 모든 파일, 하위 디렉터리에 대해서도 적용

- 사용권한 rw- rw- r--
- 2진수: 110 110 100
- 8진수: 6 6 4
- $ chmod 664 cs1.txt

- [u|g|o|a]+[+|-|=][r|w|x]+
- u(user), g(group), o(other), a(all)
- 연산자: +(추가), -(제거), =(지정)
- 권한: r(읽기), w(쓰기), x(실행)
- $ chmod g+w cs1.txt

chown(change owner)/chgrp(change group)

- chown 명령어
  파일이나 디렉터리의 소유자를 변경할 때 사용한다

  ```
  $ chown 사용자 파일
  $ chown [-R] 사용자 디렉터리
  ```

- chgrp 명령어
  파일의 그룹을 변경할 수 있다

  ```
  $ chgrp 그룹 파일
  $ chgrp [-R] 그룹 디렉터리
  ```

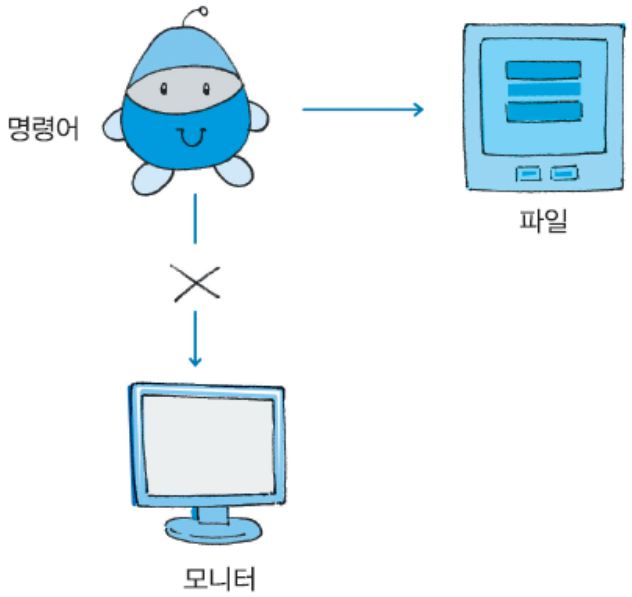
- 파일의 소유자 또한 슈퍼 유저만이 사용 가능 !

입출력 재지정 및 파이프

출력 재지정(output redirection)

- 명령어의 표준출력 내용을 모니터에 출력하는 대신에 파일에 저장

```
$ 명령어 > 파일
$ who > names.txt
```



출력 재지정 예

- $ cat > list1.txt
  Hi !
  This is the first list.
  ^D
- $ cat > list2.txt
  Hello !
  This is the second list.
  ^D
- $ cat list1.txt list2.txt > list3.txt
- $ cat list3.txt
  Hi !
  This is the first list.
  Hello !
  This is the second list.

출력 추가

- 명령어의 표준출력을 모니터 대신에 기존 파일에 추가
  $ 명령어 >> 파일

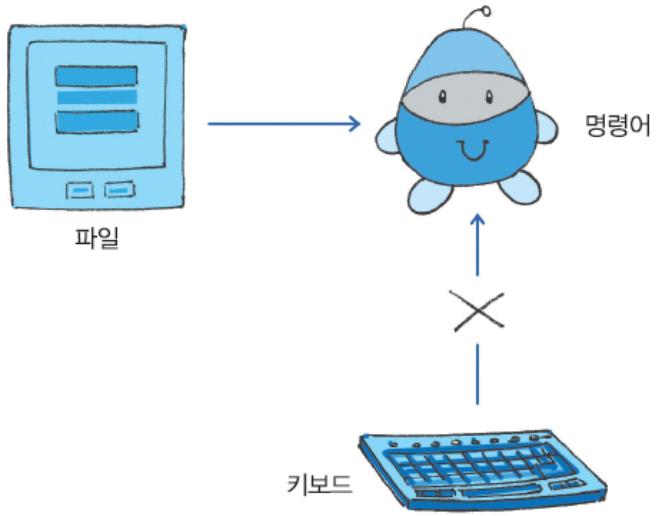
```
$ cat >> list1.txt
Bye !
This is the end of the first list.
^D

$ cat list1.txt
Hi !
This is the first list.
Bye !
This is the end of the first list.
```

입력 재지정(input redirection)

- 명령어의 표준입력을 키보드 대신에 파일에서 받는다.

```
$ 명령어 < 파일
$ wc < list1.txt
4 17 71 list1.txt
```



문서 내 입력(here document)

- 명령어의 표준입력을 단어가 다시 나타날 때까지의 내용으로
- 보통 스크립트 내에서 입력을 줄 때 사용

```
$ 명령어 << 단어
   ...
단어
```

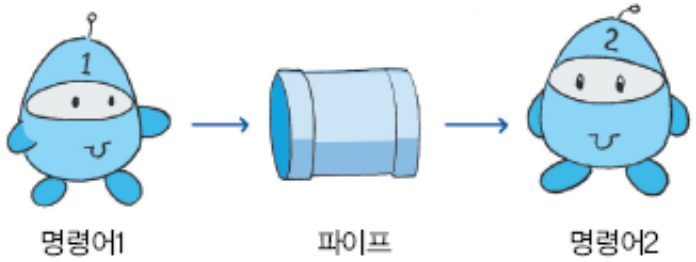
```
$ wc << end
hello !
word count
end
2 420
```

파이프

- 로그인 된 사용자들을 정렬해서 보여주기

```
$ who > names.txt
$ sort < names.txt
```

- $ 명령어1 | 명령어2
  - 명령어1의 표준출력을 명령어2의 표준입력으로 바로 받는다.



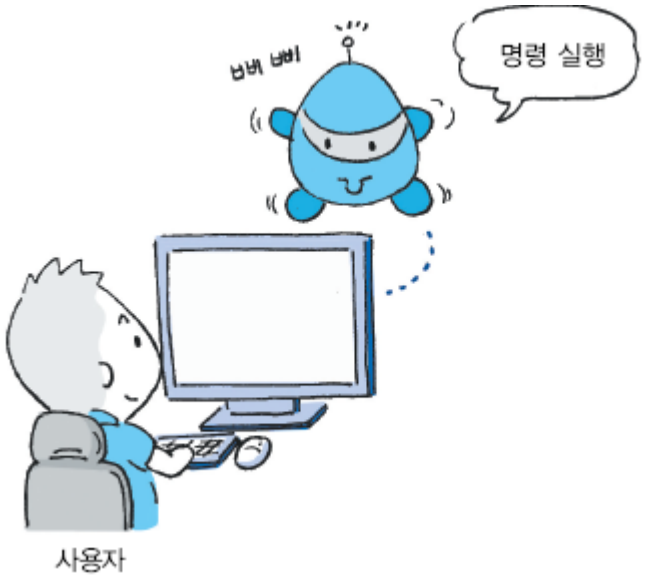

```
$ who | sort
```

후면 처리 및 프로세스

전면 처리 vs 후면처리

- 전면 처리
  - 명령어를 입력하면 명령어가 전면에서 실행되며 명령어 실행이 끝날 때까지 쉘이 기다려 준다.

- 후면 처리
  - 명령어들을 후면에서 처리하고 전면에서는 다른 작업을 할 수 있으면 동시에 여러 작업을 수행할 수 있다.
  - $ 명령어 &



후면 처리 예

- $ (sleep 100; echo done) &
  [1] 8320
- $ find . -name test.c -print &
  [2] 8325
- $ jobs
  [1] + Running (sleep 100; echo done)
  [2] - Running find . -name test.c -print

- $ fg %작업번호
  $ fg %1
  (sleep 100; echo done)

- 후면처리 입출력
  $ find . -name test.c -print > find.txt &
  $ find . -name test.c -print | mail chang &
  $ wc < inputfile &

프로세스(process)

- 실행중인 프로그램을 **프로세스**(process)라고 부른다.
- 각 프로세스는 유일한 프로세스 번호 PID를 갖는다.
- ps 명령어를 사용하여 나의 프로세스들을 볼 수 있다.

```
$ ps
PID TTY TIME CMD
8695 pts/3 00:00:00 csh
8720 pts/3 00:00:00 ps

$ ps u
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
chang 8695 0.0 0.0 5252 1728 pts/3 Ss 11:12 0:00 -csh
chang 8793 0.0 0.0 4252 940 pts/3 R+ 11:15 0:00 ps u
```

ps aux

```
$ ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.0 2064 652 ? Ss 2011 0:27 init [5]
root 2 0.0 0.0 0 0 ? S< 2011 0:01 [migration/0]
root 3 0.0 0.0 0 0 ? SN 2011 0:00 [ksoftirqd/0]
root 4 0.0 0.0 0 0 ? S< 2011 0:00 [watchdog/0]
...
root 8692 0.0 0.1 9980 2772 ? Ss 11:12 0:00 sshd: chang [pr
chang 8694 0.0 0.0 9980 1564 ? R 11:12 0:00 sshd: chang@pts
chang 8695 0.0 0.0 5252 1728 pts/3 Ss 11:12 0:00 -csh
chang 8976 0.0 0.0 4252 940 pts/3 R+ 11:24 0:00 ps aux
```

kill 명령어

- 프로세스를 강제적으로 종료시키는 명령어

  $ kill 프로세스번호

  $ kill %작업번호

  $ kill 8320 혹은

  $ kill %1

  [1] Terminated (sleep 100; echo done)