

# 시스템 프로그래밍

**Linux system programming**

**Unix/Linux 사용하기**

<https://open.kakao.com/o/gYrt5Z2d>



**KYUNGSUNG UNIVERSITY SINCE 1955**

# 간단한 명령어 사용

\$ date

\$ hostname

\$ uname

\$ who

\$ ls

\$ clear

\$ passwd



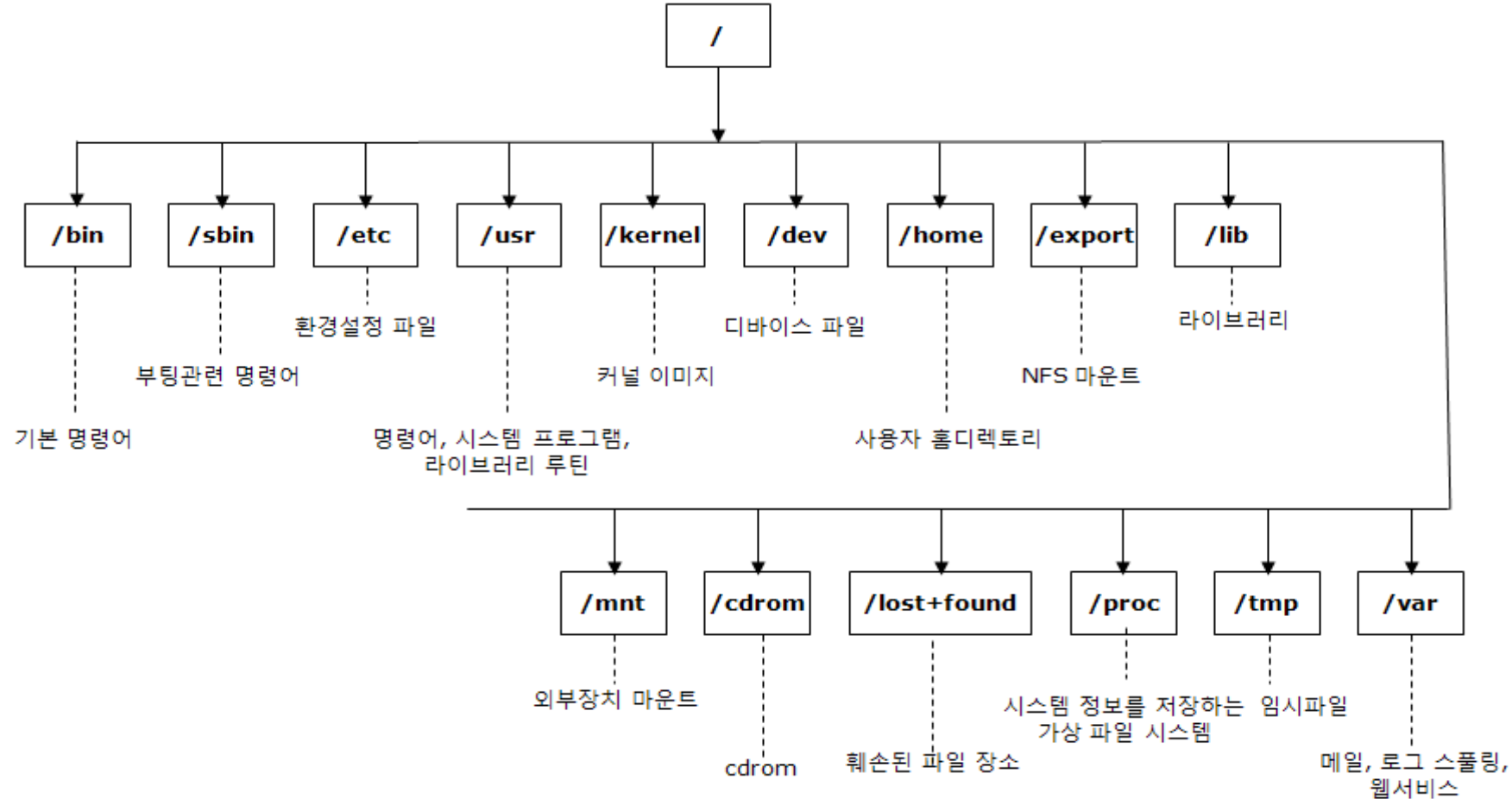
# 파일의 종류

- 일반 파일(ordinary file)
  - 데이터를 가지고 있으면서 디스크에 저장된다.
- 디렉터리(directory)/폴더(folder)
  - 디렉터리(폴더) 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



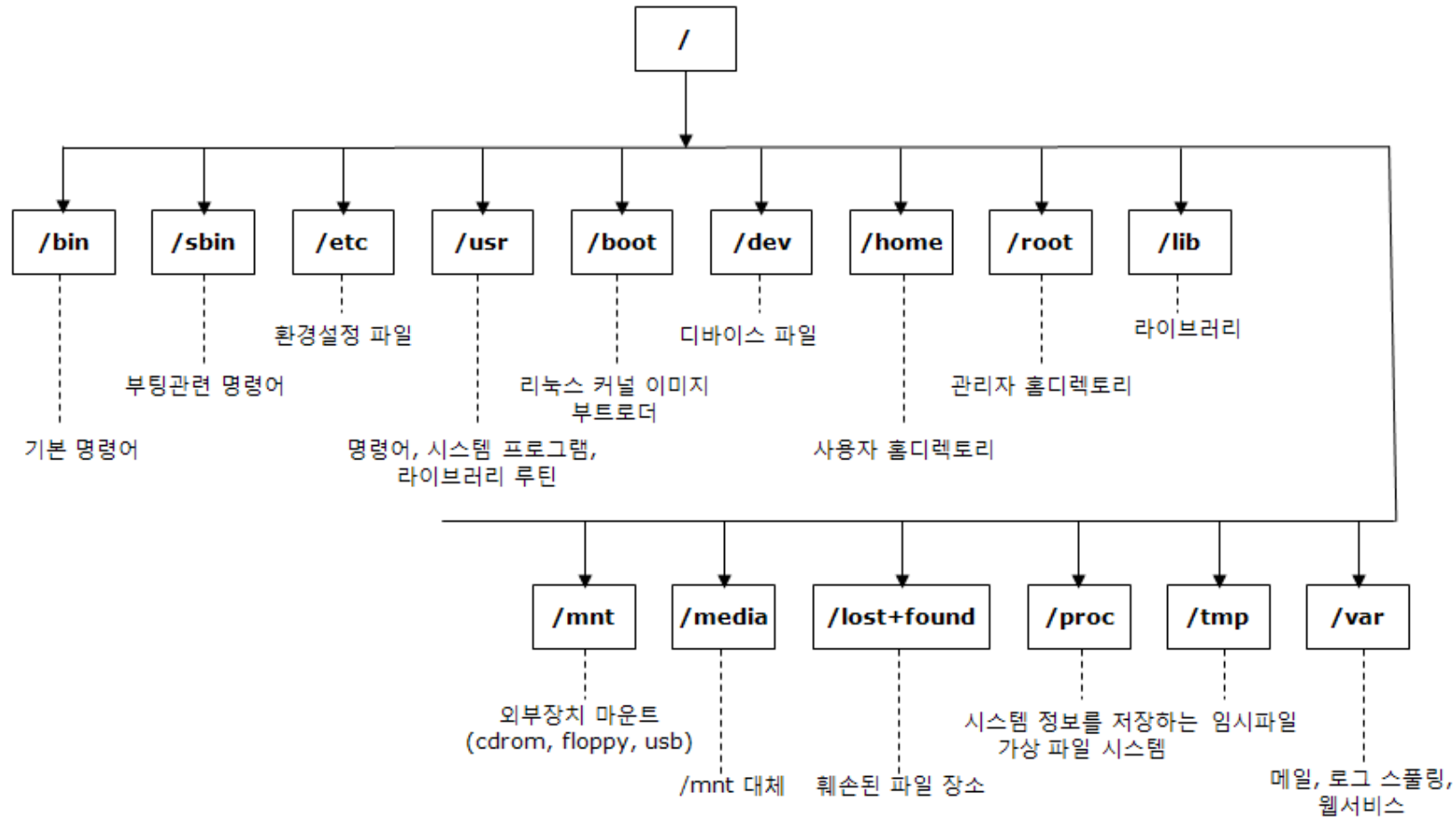
# 디렉터리 계층구조

- 유닉스의 디렉터리는 루트로 부터 시작하여 계층구조를 이룬다.



# 디렉터리 계층구조

- 리눅스 디렉터리



# 홈 디렉터리/현재 작업 디렉터리

- 홈 디렉터리(home directory)
  - 각 사용자마다 별도의 홈 디렉터리가 있음
  - 사용자가 로그인하면 홈 디렉터리에서 작업을 시작함
- 현재 작업 디렉터리(current working directory)
  - 현재 작업 중인 디렉터리
  - 로그인 하면 홈 디렉터리에서부터 작업이 시작된다.



## 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 디렉터를 만듦
  - `$ 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
- \$ 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

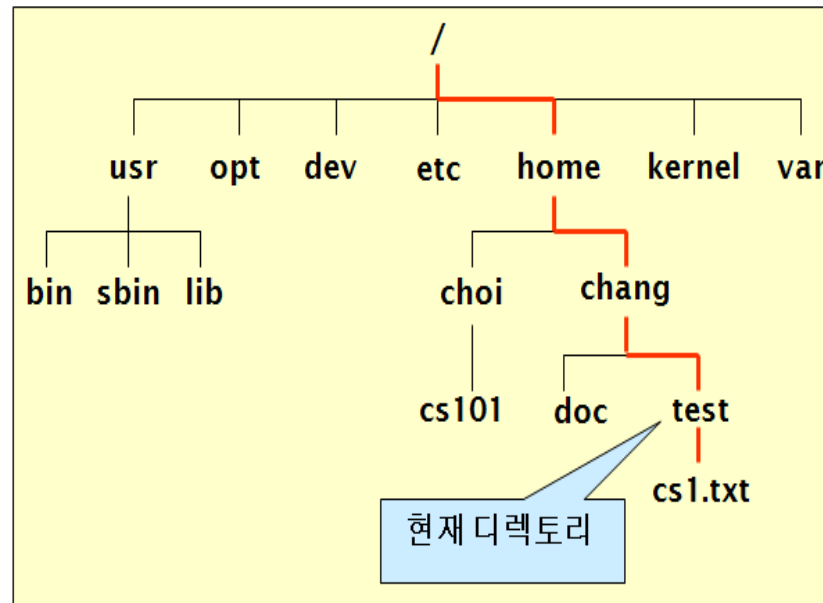
명령어	의미
ls	파일 및 디렉터리 리스트
ls -a	모든 파일과 디렉터리 리스트
ls -asl	모든 파일 자세히 리스트
mkdir	디렉터리 만들기
cd 디렉터리	디렉터리로 이동
cd	홈 디렉터리로 이동
cd ~	홈 디렉터리로 이동
cd ..	부모 디렉터리로 이동
pwd	현재 작업 디렉터리 프린트



# 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 디렉터리



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 만듦

```
$ 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 만듦

```
$ 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 <sup>+</sup>	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 <sup>+</sup>	파일 삭제
rmdir 디렉터리 <sup>+</sup>	디렉터리 삭제
grep 키워드 파일	파일에서 키워드 찾기



# 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--



# 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 !

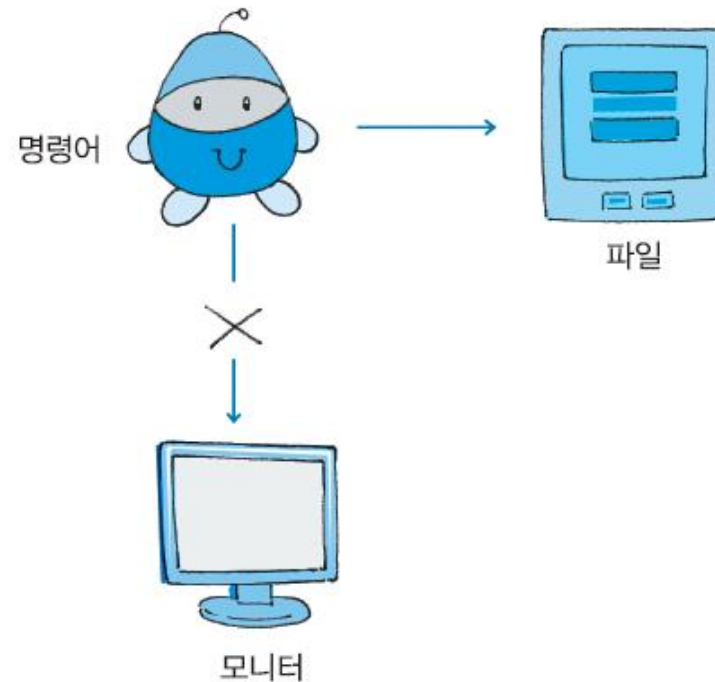


# 출력 재지정(output redirection)

- 명령어의 표준출력 내용을 모니터에 출력하는 대신에 파일에 저장

\$ 명령어 > 파일

\$ who > names.txt



## 출력 재지정 예

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





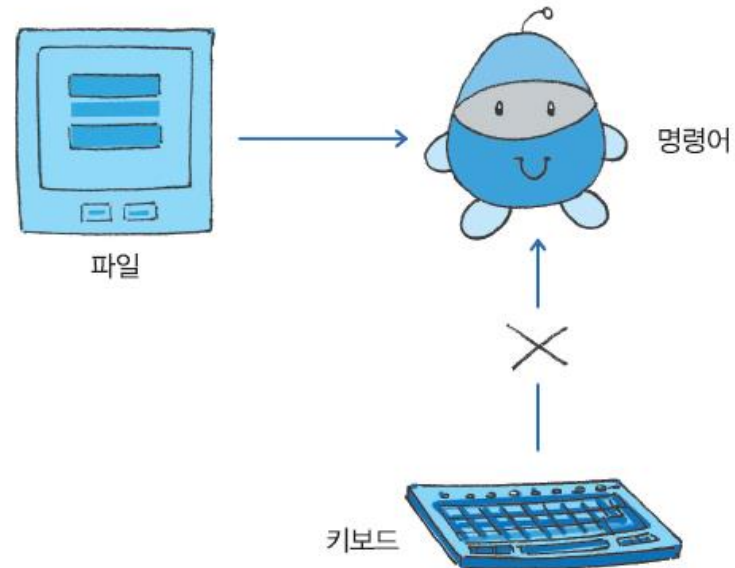
# 입력 재지정(input redirection)

- 명령어의 표준입력을 키보드 대신에 파일에서 받는다.

\$ 명령어 < 파일

\$ wc < list1.txt

4 17 71 list1.txt



## 문서 내 입력(here document)

- 명령어의 표준입력을 **단어**가 다시 나타날 때까지의 내용으로
- 보통 스크립트 내에서 입력을 줄 때 사용

\$ 명령어 << 단어

...

단어

\$ wc << end

hello !

word count

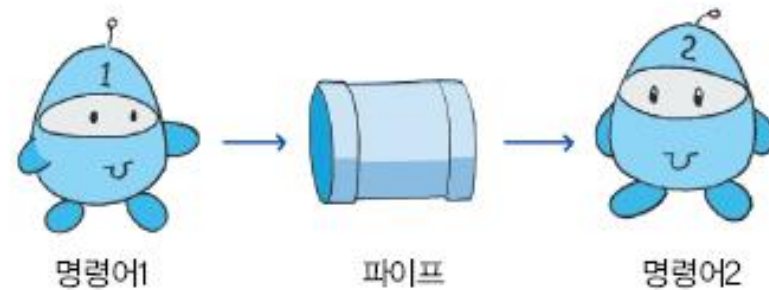
end

2 420



# 파이프

- 로그인 된 사용자들을 정렬해서 보여주기  
\$ who > names.txt  
\$ sort < names.txt
- \$ 명령어1 | 명령어2
  - 명령어1의 표준출력을 명령어2의 표준입력으로 바로 받는다.



\$ who | sort



# 전면 처리 vs 후면처리

- 전면 처리

- 명령어를 입력하면 명령어가 전면에서 실행되며 명령어 실행이 끝날 때까지 쉼이 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 강제로 종료시키는 명령어

\$ kill 프로세스번호

\$ kill %작업번호

\$ kill 8320 혹은

\$ kill %1

[1] Terminated ( sleep 100; echo done )





## Reference

- ✓ 리눅스 프로그래밍, 창병모, 생능출판
- ✓ <https://www.44bits.io/ko/post/wsl2-install-and-basic-usage>

