

# Linux 기본 명령어

---

네트워크 프로그래밍

# 파일의 종류

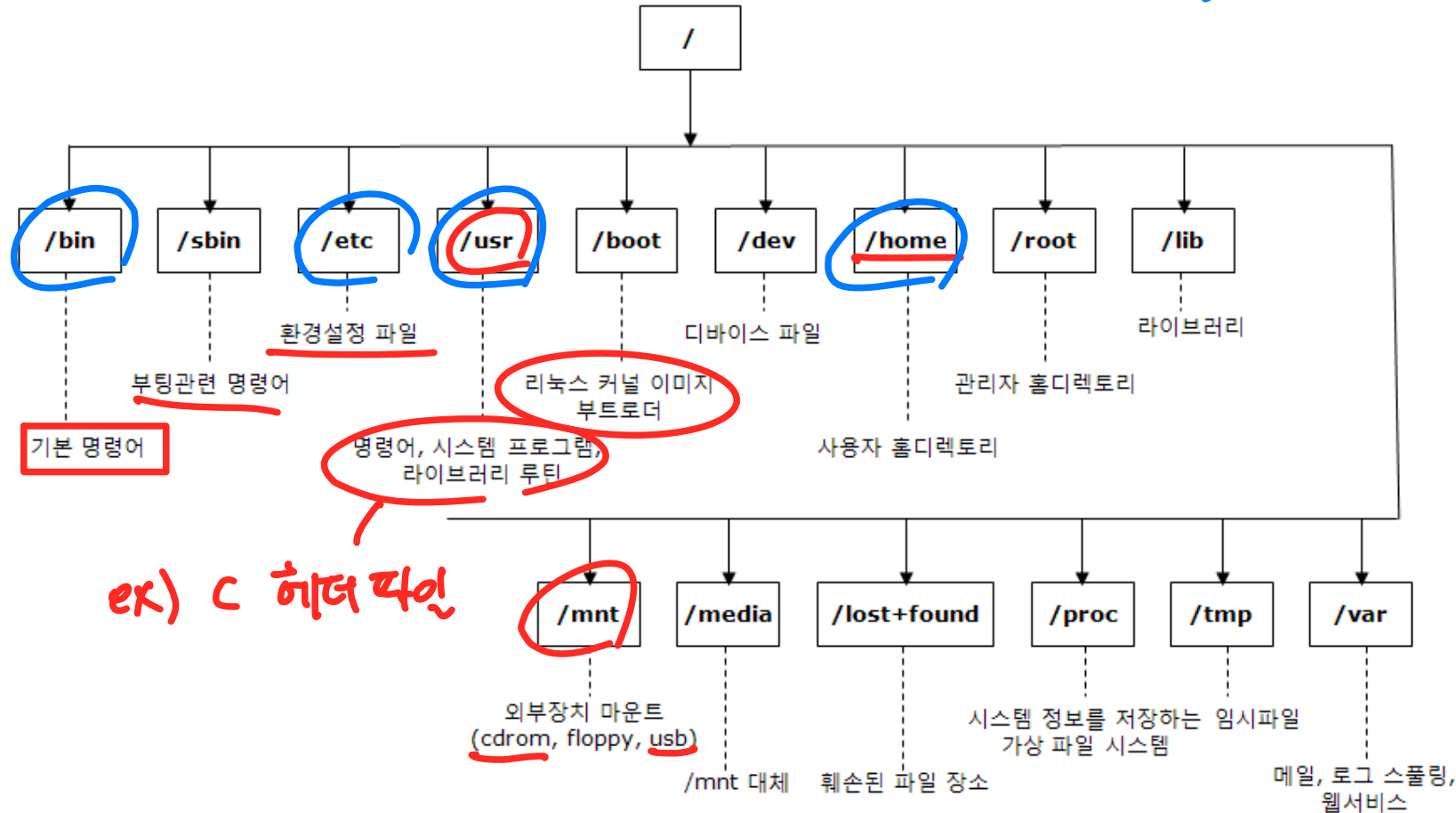
---

- 일반 파일(ordinary file)
  - 데이터를 가지고 있으면서 디스크에 저장된다.
- 디렉토리(directory) 또는 폴더(folder)
  - 디렉토리(폴더) 자체도 하나의 파일로 취급
  - 한 디렉토리는 다른 디렉토리들을 포함함으로써 계층 구조를 이룬다.
  - 부모 디렉토리는 다른 디렉토리들을 서브 디렉토리로 갖는다.
- 특수 파일(special file)
  - 물리적인 장치에 대한 내부적인 표현
  - 키보드(stdin), 모니터(stdout), 프린터 등도 파일처럼 사용
  - 소켓 : *과일*

# 디렉토리 계층구조

## ■ 리눅스 디렉토리

Q: 뭐 쓰게 될..



# 홈 디렉토리/현재 작업 디렉토리

---

- 홈 디렉토리(home directory)
  - 각 사용자마다 별도의 홈 디렉토리가 있음
  - 사용자가 로그인하면 홈 디렉토리에서 작업을 시작함
- 현재 작업 디렉토리(current working directory)
  - 현재 작업 중인 디렉토리
  - 로그인하면 홈 디렉토리에서부터 작업이 시작된다.
  - \$pwd 명령어로 현재 작업 디렉토리 확인

*print - working directory*

# 디렉토리 관련 명령

---

- pwd (print working directory)

- 현재 작업 디렉토리를 프린트

```
$ pwd
```

- cd (change directory)

- 현재 작업 디렉토리를 이동

- \$ cd 디렉토리

← windows 동일

```
$ cd temp
```

- mkdir (make directory)

- 새 디렉토리를 만듦

- \$ mkdir 디렉토리

```
$ mkdir workspace
```

temp

root 폴더로 이동 : \$ cd

# 디렉토리 리스트

## ls(list): 파일 리스트 보기

- -F: 파일 유형을 나타내는 기호를 파일명 끝에 표시
  - 디렉토리는 '/', 실행파일은 '\*', 심볼링 링크는 '@'가 표시됨
- -l: 파일에 관한 상세 정보를 나타냄
- -a: dot 파일(.cache)을 포함한 모든 파일 표시
- -t: 파일이 생성된 시간별로 표시
- -R: 서브 디렉토리의 내용까지 표시

dir

\$ ls -al

\$ ll

```
$ ls -F
PycharmProjects/  eclipse-workspace/  hello.c          temp/      문서/      사진/
anaconda3/        examples.desktop   network_workspace/  공개/      바탕화면/  음악/
eclipse/          hello*            snap/            다운로드/  비디오/   템플릿/

$ ls -al
합계 176
drwxr-xr-x 29 changsu changsu 4096 9월  3 13:14 .
drwxr-xr-x  3 root    root   4096 7월  3 06:02 ..
-rw-----  1 changsu changsu 1356 9월  2 14:32 .ICEauthority
drwxr-xr-x  4 changsu changsu 4096 7월  3 09:46 .PyCharmCE2019.1
-rw-----  1 changsu changsu 1296 9월  3 10:02 .bash_history
-rw-r--r--  1 changsu changsu  220 7월  3 06:02 .bash_logout
-rw-r--r--  1 changsu changsu 3894 7월  3 10:04 .bashrc
drwx----- 18 changsu changsu 4096 9월  2 19:10 .cache
drwx----- 13 changsu changsu 4096 7월  3 10:27 .config
drwxr-xr-x  6 changsu changsu 4096 7월  3 10:08 .eclipse
drwx-----  3 changsu changsu 4096 7월  3 10:31 .gnupg
drwxr-xr-x  4 changsu changsu 4096 7월  3 09:46 .java
```

# 와일드 카드 사용

windows: dir \* 는 파일

## ■ 와일드 카드 종류

- \*: 임의의 문자열

ls 는 ls \* 와 동일

```
$ ls *
app.txt      deref      hello.c      hello.o      neighbor     NetInt.java  pointer2     pointer3.c
cond         deref.c    hello_client hello_server.c neighborvar.c point        pointer2.c   pointer.c
condition.c  hello     hello_client.c memory_layout NetInt.class pointer      pointer3     temp
```

```
$ ls -al p*
-rwxr-xr-x 1 pi pi 6084 Mar 31 2019 point
-rwxr-xr-x 1 pi pi 6004 Mar 31 2019 pointer
-rwxr-xr-x 1 pi pi 5780 Mar 31 2019 pointer2
-rw-r--r-- 1 pi pi 300 Mar 31 2019 pointer2.c
-rwxr-xr-x 1 pi pi 6284 Mar 31 2019 pointer3
-rw-r--r-- 1 pi pi 511 Mar 31 2019 pointer3.c
-rw-r--r-- 1 pi pi 459 Mar 31 2019 pointer.c
```

p로 시작하는 모든  
파일 화면 표시

## ?: 임의의 한 문자를 의미

```
$ ls ap?.txt
app.txt

$ ls ?pp.txt
app.txt
```

# 디렉토리 관련 명령어

· / : 현재위치

| 명령어               | 의미                                      |
|-------------------|---|
| ls                | 파일 및 디렉토리 보기                            |
| ls -a             | 모든 파일과 디렉토리 보기 (간단히 보기)                 |
| ls -alF (ll)      | 실행파일은 *, 경로 /, 소켓=, 링크 @ 등의 지시자로 출력     |
| ls -al            | 모든 파일 자세히 보기 (숨긴 파일 표시)                 |
| ls -alS           | -S: 파일 크기 순으로 정렬하여 보기                   |
| cd 디렉토리           | 디렉토리로 이동                                |
| cd                | 어떤 위치에서도 홈 디렉토리로 이동                     |
| cd ..             | 현재 디렉토리의 <u>한 단계 상위 디렉토리로 이동</u>        |
| cd /              | 최상위 루트(/) 디렉토리로 이동                      |
| pwd               | 현재 작업 디렉토리 출력 (print working directory) |
| <u>rmdir 디렉토리</u> | 디렉토리 삭제                                 |
| <u>mkdir 디렉토리</u> | 디렉토리 만들기                                |



# 경로명

- 파일이나 디렉토리에 대한 정확한 이름

*\$/ server*

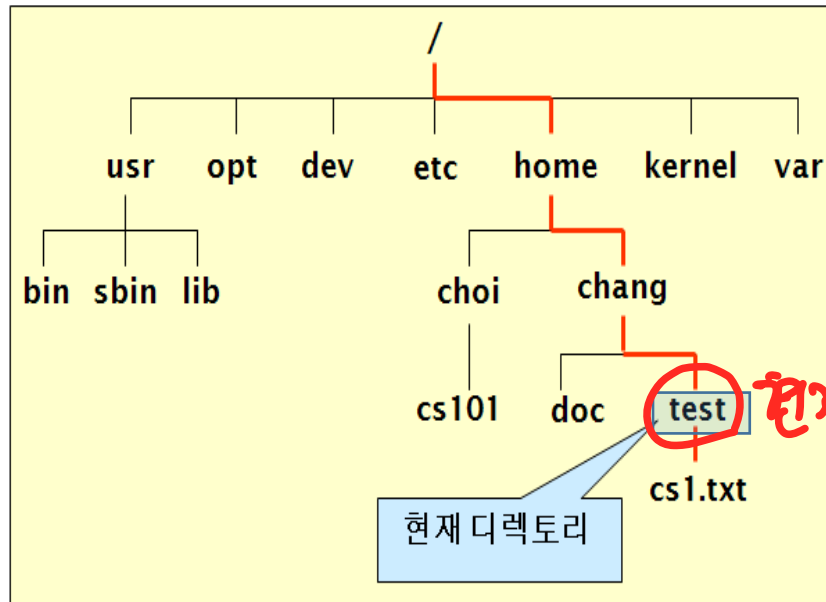
- 절대 경로명 (absolute pathname)

- 루트 디렉토리로부터 시작하여 경로 이름을 정확하게 적는 것

*: 모든 경로*

- 상대 경로명 (relative path name)

- 현재 작업 디렉토리부터 시작해서 경로 이름을 적는 것



~ : 홈 디렉토리  
.: 현재 디렉토리  
..: 부모 디렉토리

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

cs1.txt의 상대 경로명  
`cs1.txt`

test폴더에서 doc폴더로 이동

*상위*  
↓

`$cd ../doc`

# 파일 내용 보기: cat 명령어

---

## ■ 파일 내용 출력

- \$ cat 파일이름

```
$ cat hello.c
#include <stdio.h>

int main()
{
    printf("Hello World\n");

    return 0;
}
```

## ■ 파일 생성

- \$ cat > 파일이름

```
$ cat > test.txt
ABCDEFGHIJKLMN
^D

$ cat test.txt
ABCDEFGHIJKLMN
```

# 파일복사: cp 명령어

## ■ 파일 복사

### • \$cp 파일1 파일2

- 파일1의 복사본을 파일2의 이름으로 현재 디렉토리에 복사

```
$ cp cs1.txt cs2.txt  
$ ls -al  
-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
```

파일 크기가 같아야함.

### • \$cp 파일1 디렉토리

- 파일1의 복사본을 디렉토리(data) 내부로 복사함

```
$ cp cs1.txt data/
```

cp dir.txt temp/

하위 temp 폴더에 저장.

# 파일 이동: mv 명령어

삭제.

## ■ 파일 이름 변경

- \$mv 파일1 파일2
- 파일1의 이름을 파일2로 변경
- 리눅스는 rename 명령어가 없음 (mv 명령어를 대신 사용)

```
$ mv cs2.txt cs3.txt
```

\$mv dir1.txt dir2.txt

## ■ 파일을 특정 디렉토리(data)로 이동

- \$ mv 파일명 디렉토리

```
$ mv cs3.txt data/
```

\$ mv dir2.txt temp/

⇒ 파일 이름 변경.

① 파일 삭제. → EOF

⇓  
② 파일 저장

→ 3) 기존 파일 삭제.

# 파일/디렉토리 삭제: rm

## ■ 파일 삭제

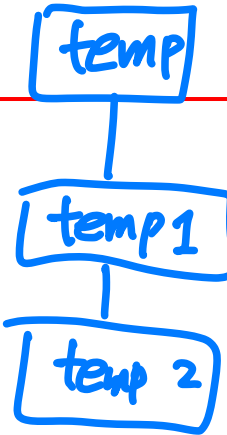
- \$rm 파일이름
- 해당 파일을 지움

```
$rm cs1.txt
```

```
$ mkdir temp
```

```
$ cd temp
```

```
$ mkdir temp2
```



## ■ 디렉토리 삭제

- \$ rm -r 디렉토리
- 디렉토리 내의 모든 파일 및 하위 디렉토리들을 강제로 지움

*rm -rf.*

```
$ cd ../..
```

```
$ rm -r temp1
```

## ■ 디렉토리 삭제: rmdir(remove directory)

- \$rmdir 디렉토리

-디렉토리 내부에 아무 파일도 없어야 삭제가 됨

*이름칠때 tab 키..!*

```
$ ls temp  
1.txt 2.txt
```

temp 디렉토리 내부에  
1.txt, 2.txt 파일이  
존재하는 경우

```
$ rmdir temp  
rmdir: failed to remove 'temp': Directory not empty
```

```
$ rm -r temp
```

temp 디렉토리 강제 삭제됨

# 파일에서 키워드 찾기: grep #1

## ■ 테스트용 파일 생성

```
$ man head
```

➤ head 명령어: 파일의 앞 부분을 확인

## • head 명령어의 메뉴얼을 head.txt 파일로 저장

```
$ man head > head.txt
```

```
$ vi head.txt
```

```
changsu@dell-d08: ~/temp/data
1 HEAD(1) User Commands HEAD(1)
2
3 NAME
4 head - output the first part of files
5
6 SYNOPSIS
7 head [OPTION]... [FILE]...
8
9 DESCRIPTION
10 Print the first 10 lines of each FILE to standard output.
11 With more than one FILE, precede each with a header giving the
12 file name.
13
14 With no FILE, or when FILE is -, read standard input.
15
16 Mandatory arguments to long options are mandatory for short
17 options too.
```

```
changsu@dell-d08: ~/temp/data
HEAD(1) User Commands HEAD(1)

NAME
head - output the first part of files

SYNOPSIS
head [OPTION]... [FILE]...

DESCRIPTION
Print the first 10 lines of each FILE to standard output.
With more than one FILE, precede each with a header giving the
file name.

With no FILE, or when FILE is -, read standard input.

Mandatory arguments to long options are mandatory for short
options too.

-c, --bytes=[-]NUM
    print the first NUM bytes of each file; with the lead-
    ing '-', print all but the last NUM bytes of each file

-n, --lines=[-]NUM
    print the first NUM lines instead of the first 10; with
    the leading '-', print all but the last NUM lines of

Manual page head(1) line 1 (press h for help or q to quit)
```

# 파일에서 키워드 찾기: grep

## ■ grep 명령어

*검색기*

- `$ grep [옵션] [키워드] [파일명]`
- 옵션 `-n`: 파일 내에서 행 번호와 검색된 단어(결과)를 같이 출력

*line number*

```
$ grep -n first head.txt
4:      head - output the first part of files
10:     Print the first 10 lines of each FILE to standard output.
20:     print the first NUM bytes of each file; with the lead-
24:     print the first NUM lines instead of the first 10; with
```

- 옵션 `-c`: 문자열이 포함된 라인의 개수를 출력

```
$ grep -c first head.txt
4
```

- 옵션 `-i`: 대소문자를 구분하지 않고 키워드 검색

*ignore 대소문자.*

```
$ grep -i print head.txt
Print the first 10 lines of each FILE to standard output.
print the first NUM bytes of each file; with the lead-
ing '-', print all but the last NUM bytes of each file
print the first NUM lines instead of the first 10; with
the leading '-', print all but the last NUM lines of
never print headers giving file names
always print headers giving file names
```

# 파일 관련 명령어

---

| 명령어            | 의미                             |
|----------------|--------------------------------|
| cat 파일         | 파일 내용 출력                       |
| more           | 한 번에 한 페이지씩 화면에 표시             |
| head 파일        | 파일의 앞부분 출력 (\$head -n 3 1.txt) |
| tail 파일        | 파일의 뒷 부분 출력                    |
| wc 파일          | 줄/단어/문자수 카운트                   |
| cp 파일1 파일2     | 파일1을 파일2로 복사                   |
| mv 파일1 파일2     | 파일1을 파일2로 이름 변경 및 이동           |
| rm 파일          | 파일 삭제                          |
| rmdir 디렉토리     | 디렉토리 삭제                        |
| grep 옵션 키워드 파일 | 파일에서 키워드 찾기                    |



# 파일 속성(file attribute)

- 파일의 이름, 타입, 크기, 소유자, 사용권한, 수정 시간

```
changsu@ubuntu:~/temp$ ls -als
total 44
4 drwxrwxr-x 3 changsu changsu 4096 Aug 31 10:54 .
4 drwxr-xr-x 24 changsu changsu 4096 Aug 31 10:54 ..
4 -rw-rw-r-- 1 changsu changsu 1334 Aug 31 10:50 1.txt
4 -rw-rw-r-- 1 changsu changsu 1334 Aug 31 09:50 head.txt
20 -rwxrwxr-x 1 changsu changsu 16696 Aug 31 10:54 hello
4 -rw-rw-r-- 1 changsu changsu 73 Aug 31 10:54 hello.c
4 drwxrwxr-x 2 changsu changsu 4096 Aug 31 10:49 test
```

- -s: size(파일의 블록 수)

| 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    |

## ■ 파일 사용권한

- 소유자(owner)/그룹(group)/기타(others)로 구분하여 관리

rwX

rwX

rwX

| 소유자  | 그룹  | 기타  | 소유자       | 소유그룹     |                            |
|------|-----|-----|-----------|----------|----------------------------|
| -rwX | rwX | r-x | 1 changsu | changsus | 16696 Aug 31 10:54 hello   |
| -rw- | rw- | r-- | 1 changsu | changsus | 1334 Aug 31 09:50 head.txt |

# chmod(change mode) 명령어 #1

## ■ chmod 명령어

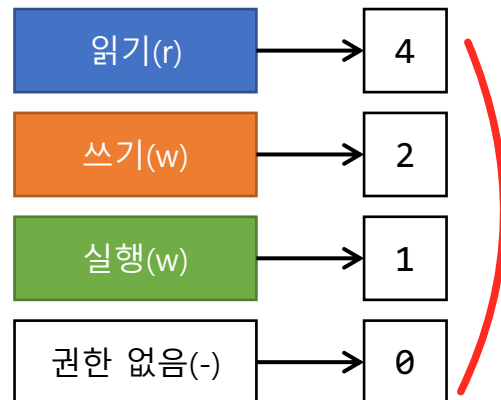
- 파일 혹은 디렉토리의 사용권한을 변경하는 명령어
- `chmod [option] [mode] [file]`

• -R 옵션  $\Rightarrow$  *Recursive*

-지정한 모드를 파일과 디렉토리에 대해 재귀적으로 적용(하위 폴더 같이 적용)

## • mode

- u (소유자), g (그룹), o (그외 사용자), a (모든 사용자)
- + (권한 추가), - (권한 제거), = (현재 모드로 권한 지정)
- *r* (읽기 권한), *w* (쓰기 권한), *x* (실행 권한)
- 8진수 형태로 권한 추가 가능



|     |   |   |         |
|-----|---|---|---------|
| --- | : | 0 | (0+0+0) |
| r-- | : | 4 | (4+0+0) |
| -w- | : | 2 | (0+2+0) |
| --x | : | 1 | (0+0+1) |
| rw- | : | 6 | (4+2+0) |
| r-x | : | 5 | (4+0+1) |
| -wx | : | 3 | (0+2+1) |
| rxw | : | 7 | (4+2+1) |

# chmod(change mode) 명령어 #2

## ■ 초기 파일 권한 상태

```
changsu@ubuntu:~/temp$ ll
total 44
drwxrwxr-x  3 changsu changsu  4096 Aug 31 10:54 ./
drwxr-xr-x 24 changsu changsu  4096 Aug 31 10:54 ../
-rw-rw-r--  1 changsu changsu  1334 Aug 31 10:50 1.txt
-rw-rw-r--  1 changsu changsu  1334 Aug 31 09:50 head.txt
-rwxrwxr-x  1 changsu changsu 16696 Aug 31 10:54 hello*
-rw-rw-r--  1 changsu changsu    73 Aug 31 10:54 hello.c
drwxrwxr-x  2 changsu changsu  4096 Aug 31 10:49 test/
```

## ■ 권한 변경 예제

- 모든 사용자가 읽고 쓰고 실행

-\$ chmod 777 파일이름

```
changsu@ubuntu:~/temp$ chmod 777 1.txt
changsu@ubuntu:~/temp$ ll 1.txt
-rwxrwxrwx 1 changsu changsu 1334 Aug 31 10:50 1.txt*
```

- 소유자는 모든 권한, 그룹 및 그 외 사용자는 읽기와 실행만 가능

-\$ chmod 755 파일이름

```
changsu@ubuntu:~/temp$ chmod 755 1.txt
changsu@ubuntu:~/temp$ ll 1.txt
-rwxr-xr-x 1 changsu changsu 1334 Aug 31 10:50 1.txt*
```

# chmod(change mode) 명령어 #3

- 실행(x) 권한 제거

-\$ chmod -x 파일이름

```
changsu@ubuntu:~/temp$ ll 1.txt
-rwxr-xr-x 1 changsu changsu 1334 Aug 31 10:50 1.txt*
changsu@ubuntu:~/temp$ chmod -x 1.txt
changsu@ubuntu:~/temp$ ll 1.txt
-rw-r--r-- 1 changsu changsu 1334 Aug 31 10:50 1.txt
```

- 실행(x) 권한 추가

-\$ chmod +x 파일이름

```
changsu@ubuntu:~/temp$ chmod +x 1.txt
changsu@ubuntu:~/temp$ ll 1.txt
-rwxr-xr-x 1 changsu changsu 1334 Aug 31 10:50 1.txt*
```

- 다른 사용 예제

*user + read, write.*

-파일 소유자의 읽기, 쓰기 권한 추가

-\$ chmod u+rw 파일이름

```
changsu@ubuntu:~/temp$ chmod 000 1.txt
changsu@ubuntu:~/temp$ ll 1.txt
----- 1 changsu changsu 1334 Aug 31 10:50 1.txt
changsu@ubuntu:~/temp$ chmod u+rw 1.txt
changsu@ubuntu:~/temp$ ll 1.txt
-rw----- 1 changsu changsu 1334 Aug 31 10:50 1.txt
```

# chown 명령어

## ■ chown 명령어

- 파일이나 디렉토리의 소유자를 변경
  - \$ chown 소유자이름 파일이름
  - \$ chown -R 소유자이름 디렉토리

*Sudo : 관리자 권한*

```
changsu@ubuntu:~/temp$ sudo chown root 1.txt
[sudo] password for changsu:
changsu@ubuntu:~/temp$ ll 1.txt
-rw----- 1 root changsu 1334 Aug 31 10:50 1.txt
```

## ■ chgrp 명령어

- 파일의 그룹을 변경
  - \$ chgrp 그룹이름 파일이름
  - \$ chgrp -R 그룹이름 디렉토리

```
changsu@ubuntu:~/temp$ sudo chgrp root 1.txt
changsu@ubuntu:~/temp$ ll 1.txt
-rw----- 1 root root 1334 Aug 31 10:50 1.txt
```

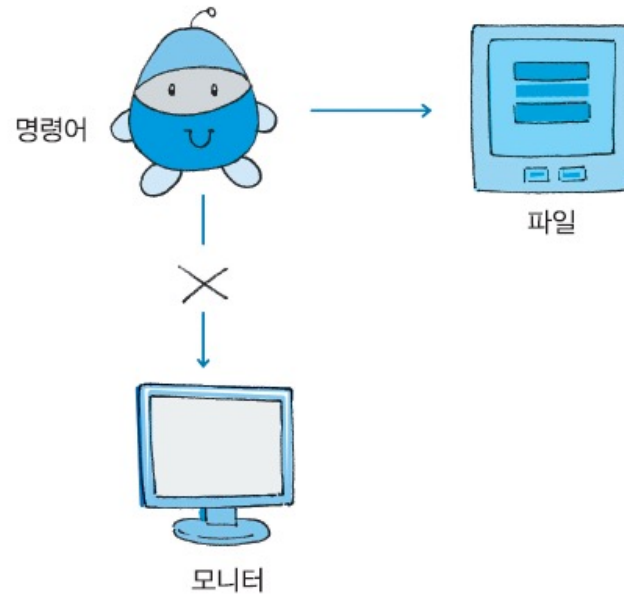
- 파일의 소유자 또한 슈퍼 유저만이 사용 가능 !

# 출력 재지정(output redirection)

- 명령어의 표준출력 내용을 모니터에 출력하는 대신에 파일에 저장

\$ 명령어 > 파일이름

\$ 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)

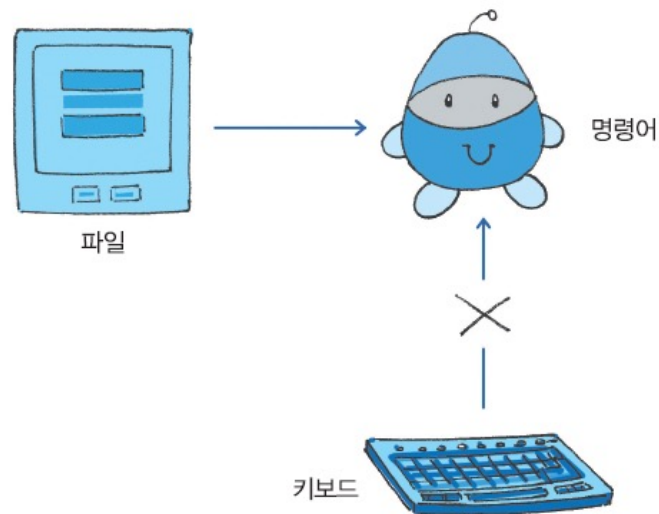
- 명령어의 표준입력을 키보드 대신에 파일에서 받는다.

\$ 명령어 < 파일

\$ wc < list1.txt

4 17 71 list1.txt

WC: word count :



# 파이프

- 로그인 된 사용자들을 정렬해서 보여주기

```
$ who > names.txt
```

```
$ sort < names.txt
```

who: 켄서 정복자

- \$ 명령어1 | 명령어2

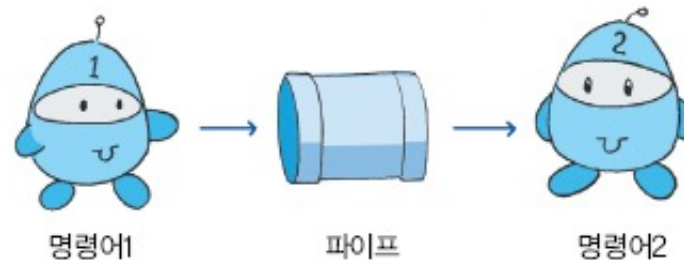
- 명령어1의 표준출력을 명령어2의 표준입력으로 바로 받는다.

\$ who | sort

파이프

↓

출력 → 입력



# 전면 처리 vs 후면처리

## ■전면 처리

- 명령어를 입력하면 명령어가 전면에서 실행되며 **명령어 실행이 끝날 때까지 쉼이 기다려 준다.**

## ■후면 처리 (background)

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

background에 m 동작~



# 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)

---

- 프로세스: 실행 중인 프로그램
- 각 프로세스는 유일한 프로세스 번호 PID를 가짐
- ps 명령어를 사용하여 실행 중인 프로세스들을 확인

\$ ps

PID TTY TIME CMD

8695 pts/3 00:00:00 csh

8720 pts/3 00:00:00 ps

\$ ps u: 특정(현재) 사용자의 프로세스 목록을 보여줌

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 : 시스템에 동작중인 모든 프로세스 확인

```
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
```

\$ ps aux | grep bash

```
changsu@ubuntu:~/temp$ ps aux | grep bash
changsu 8182 0.0 0.2 11012 5416 pts/0 Ss 10:20 0:00 bash
changsu 9081 0.0 0.0 8908 668 pts/0 S+ 12:34 0:00 grep --color=a
uto bash
```

# kill 명령어

---

- 프로세스를 강제적으로 종료시키는 명령어

\$ kill 프로세스번호

\$ kill %작업번호

\$ kill 8320 혹은

\$ kill %1

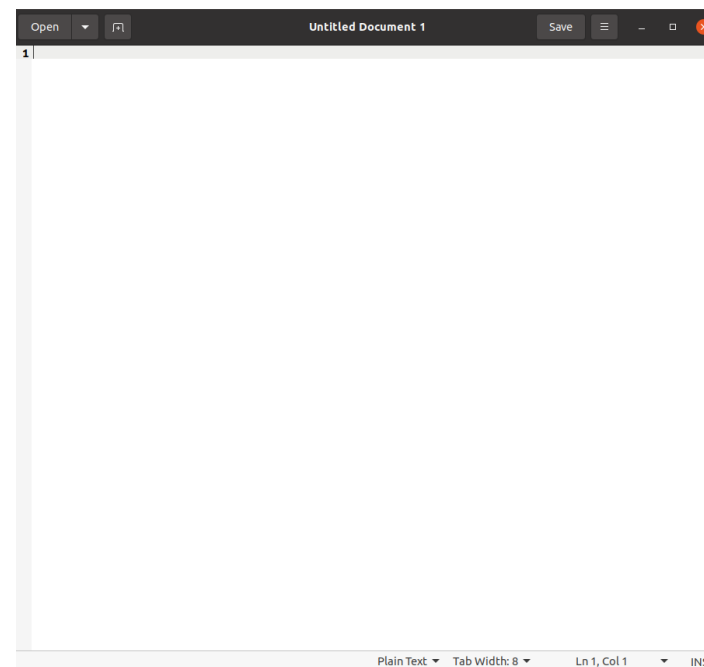
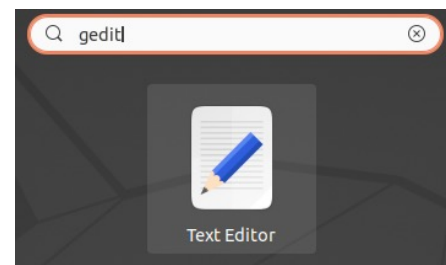
[1] Terminated ( sleep 100; echo done )



# gedit : GUI

NaNo : 기본 에디터 (터미널 상에서)

- GNOME이 제공하는 GUI 기반 문서편집기
- 사용방법
  - [프로그램] -> [보조 프로그램] -> [gedit 텍스트 편집기]
  - \$ gedit [파일이름] &
- 기능
  - 파일: 새로 만들기, 열기, 저장 닫기
  - 편집: 입력취소, 잘라내기, 복사 붙여넣기
  - 보기: 도구모음, 상태표시줄, 전체화면
  - 검색: 찾기, 바꾸기
  - 검사: 맞춤법 검사
  - 문서: 모두 저장, 모두 닫기
  - 도움말



# vim 설치하기 (Vi IMproved)

- \$ sudo apt install vim 실행

```
ubuntu-virtual-machine:~$ sudo apt install vim
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  vim-runtime
Suggested packages:
  ctags vim-doc vim-scripts
The following NEW packages will be installed:
  vim vim-runtime
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 7,111 kB of archives.
After this operation, 34.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://kr.archive.ubuntu.com/ubuntu focal/main amd64 vim-runtime all 2:8.1.2269-1ubuntu5 [5,873 kB]
Get:2 http://kr.archive.ubuntu.com/ubuntu focal/main amd64 vim amd64 2:8.1.2269-1ubuntu5 [1,238 kB]
Fetched 7,111 kB in 32s (224 kB/s)
Selecting previously unselected package vim-runtime.
(Reading database ... 151153 files and directories currently installed.)
Preparing to unpack .../vim-runtime_2%3a8.1.2269-1ubuntu5_all.deb ...
Adding 'diversion of /usr/share/vim/vim81/doc/help.txt to /usr/share/vim/vim81/doc/help.txt.vim-tiny by vim-runtime'
Adding 'diversion of /usr/share/vim/vim81/doc/tags to /usr/share/vim/vim81/doc/tags.vim-tiny by vim-runtime'
Unpacking vim-runtime (2:8.1.2269-1ubuntu5) ...
Selecting previously unselected package vim.
Preparing to unpack .../vim_2%3a8.1.2269-1ubuntu5_amd64.deb ...
Unpacking vim (2:8.1.2269-1ubuntu5) ...
Setting up vim-runtime (2:8.1.2269-1ubuntu5) ...
Setting up vim (2:8.1.2269-1ubuntu5) ...
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vim (vim) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vimdiff (vimdiff) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rvim (rvim) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rview (rview) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vi (vi) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/view (view) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/ex (ex) in auto mode
Processing triggers for man-db (2.9.1-1) ...
```

# vim 설정하기

- `/etc/vim/vimrc` 파일을 자신의 폴더에 `.vimrc`파일로 복사하기
  - vim 설정 파일

`$ cp /etc/vim/vimrc ~/.vimrc`

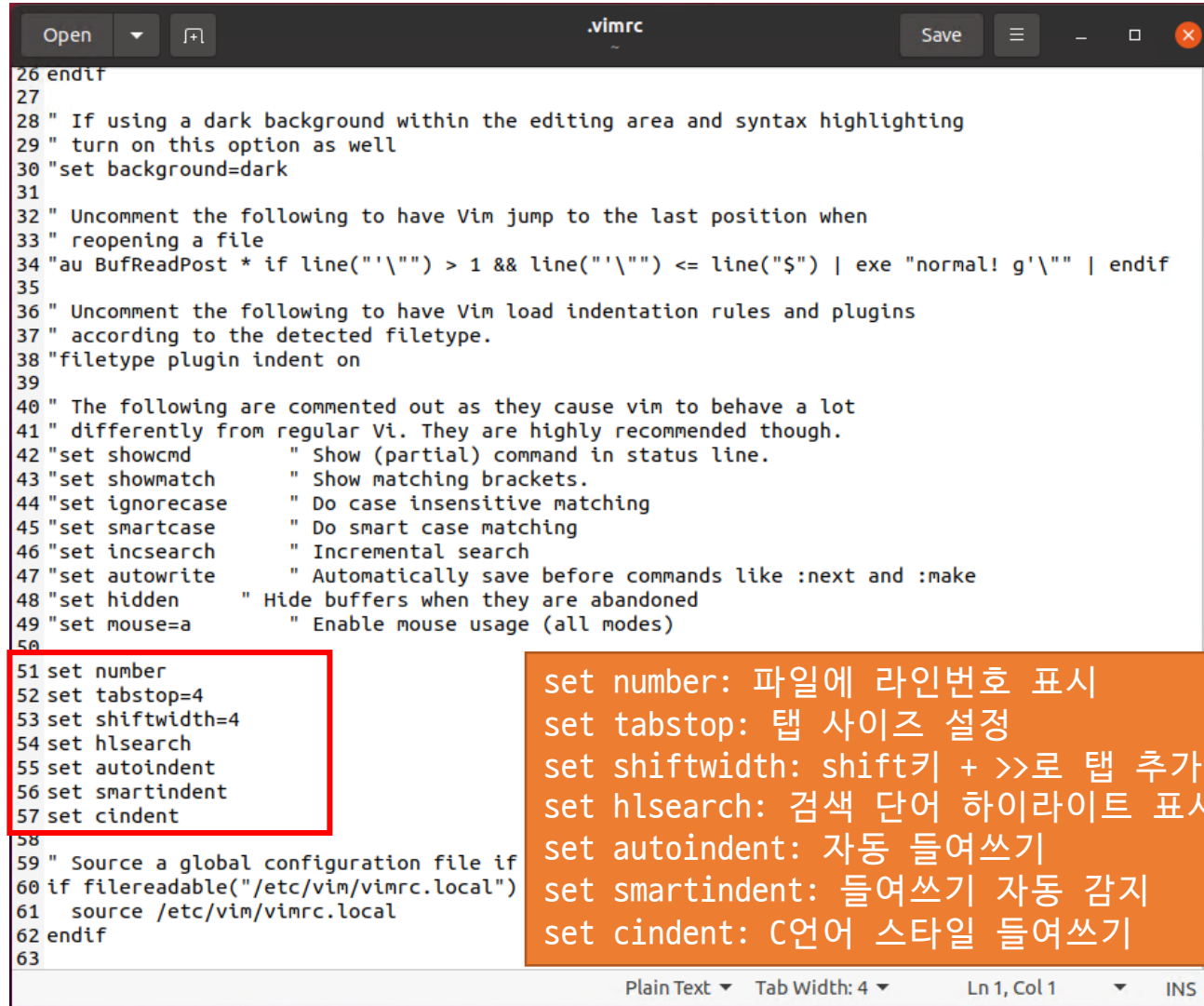
→ 내 홈 디렉토리.

- 전체 파일 확인하기: `$ ls -al`

```
changsu@changsu-virtual-machine:~$ ls -al
total 80
drwxr-xr-x 16 changsu changsu 4096 8월 30 15:34 .
drwxr-xr-x  3 root    root    4096 8월 30 14:39 ..
-rw-r--r--  1 changsu changsu  220 8월 30 14:39 .bash_logout
-rw-r--r--  1 changsu changsu 3771 8월 30 14:39 .bashrc
drwx----- 10 changsu changsu 4096 8월 30 15:03 .cache
drwx----- 12 changsu changsu 4096 8월 30 15:32 .config
drwxr-xr-x  2 changsu changsu 4096 8월 30 14:59 Desktop
drwxr-xr-x  2 changsu changsu 4096 8월 30 14:59 Documents
drwxr-xr-x  2 changsu changsu 4096 8월 30 14:59 Downloads
drwx-----  3 changsu changsu 4096 8월 30 15:15 .gnupg
drwxr-xr-x  3 changsu changsu 4096 8월 30 14:59 .local
drwxr-xr-x  2 changsu changsu 4096 8월 30 14:59 Music
drwxr-xr-x  2 changsu changsu 4096 8월 30 14:59 Pictures
-rw-r--r--  1 changsu changsu   807 8월 30 14:39 .profile
drwxr-xr-x  2 changsu changsu 4096 8월 30 14:59 Public
drwx-----  2 changsu changsu 4096 8월 30 15:15 .ssh
-rw-r--r--  1 changsu changsu     0 8월 30 15:22 .sudo_as_admin_successful
drwxr-xr-x  2 changsu changsu 4096 8월 30 14:59 Templates
drwxr-xr-x  2 changsu changsu 4096 8월 30 14:59 Videos
-rw-r--r--  1 changsu changsu 2266 8월 30 15:34 .vimrc
drwxrwxr-x  2 changsu changsu 4096 8월 30 15:31 workspace
```

# .vimrc 파일 수정

- TextEdit에서 51라인부터 57라인까지 추가



```
26 endif
27
28 " If using a dark background within the editing area and syntax highlighting
29 " turn on this option as well
30 "set background=dark
31
32 " Uncomment the following to have Vim jump to the last position when
33 " reopening a file
34 "au BufReadPost * if line("\n") > 1 && line("\n") <= line("$") | exe "normal! g'\n" | endif
35
36 " Uncomment the following to have Vim load indentation rules and plugins
37 " according to the detected filetype.
38 "filetype plugin indent on
39
40 " The following are commented out as they cause vim to behave a lot
41 " differently from regular Vi. They are highly recommended though.
42 "set showcmd      " Show (partial) command in status line.
43 "set showmatch    " Show matching brackets.
44 "set ignorecase   " Do case insensitive matching
45 "set smartcase    " Do smart case matching
46 "set incsearch    " Incremental search
47 "set autowrite    " Automatically save before commands like :next and :make
48 "set hidden      " Hide buffers when they are abandoned
49 "set mouse=a     " Enable mouse usage (all modes)
50
51 set number
52 set tabstop=4
53 set shiftwidth=4
54 set hlsearch
55 set autoindent
56 set smartindent
57 set cindent
58
59 " Source a global configuration file if
60 if filereadable("/etc/vim/vimrc.local")
61   source /etc/vim/vimrc.local
62 endif
63
```

Handwritten red text: `:wq =>` 저장 후 종료

Orange box text:

- set number: 파일에 라인번호 표시
- set tabstop: 탭 사이즈 설정
- set shiftwidth: shift키 + >>로 탭 추가할 때
- set hlsearch: 검색 단어 하이라이트 표시
- set autoindent: 자동 들여쓰기
- set smartindent: 들여쓰기 자동 감지
- set cindent: C언어 스타일 들여쓰기

# .bashrc 파일에 vim 연결

i: ↓

- vi 명령어로 vim 프로그램 실행
  - alias vi='vim' 내용 추가

91: alias ll='ls -aLF'  
=wq

```
70 ;;
71 *)
72 ;;
73 esac
74
75 # enable color support of ls and also add handy aliases
76 if [ -x /usr/bin/dircolors ]; then
77     test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
78     alias ls='ls --color=auto'
79     #alias dir='dir --color=auto'
80     #alias vdir='vdir --color=auto'
81
82     alias grep='grep --color=auto'
83     alias fgrep='fgrep --color=auto'
84     alias egrep='egrep --color=auto'
85 fi
86
87 # colored GCC warnings and errors
88 #export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:quote=01'
89
90 # some more ls aliases
91 alias ll='ls -aLF'
92 alias la='ls -A'
93 alias l='ls -CF'
94
95 alias vi='vim'
96 # Add an "alert" alias for long running commands. Use like so:
97 # sleep 10; alert
98 alias alert='notify-send --urgency=low -i "${[ $? = 0 ] && echo terminal || echo error}" "$(his
    tory|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\`)"'
99
100 # Alias definitions.
101 # You may want to put all your additions into a separate file like
102 # ~/.bash_aliases, instead of adding them here directly.
103 # See /usr/share/doc/bash-doc/examples in the bash-doc package.
104
105 if [ -f ~/.bash_aliases ]; then
```

96,1

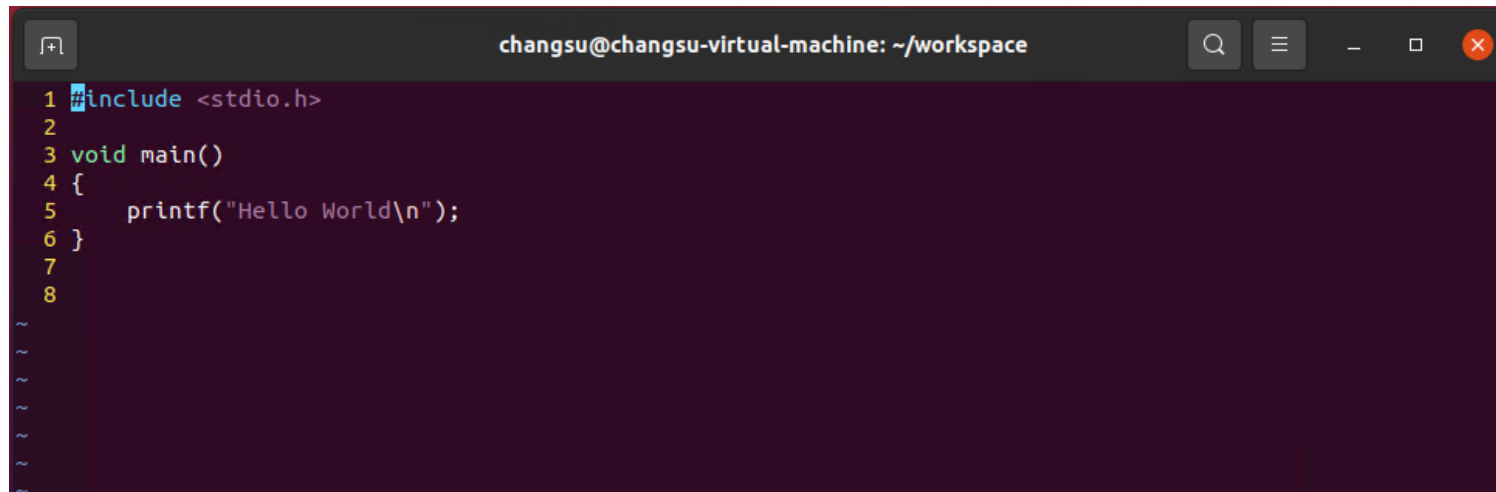
70%

# 수정된 .bashrc 파일 적용하기

- 파일 수정 이후 수정된 파일을 시스템에 적용
  - \$ source ~/.bashrc
    - source: 수정된 파일 적용
- 변경 내용
  - C 문법에 따라 다른 색상 표시
  - 라인 번호 표시
  - 자동 들여쓰기 등

바로 적용이 안되네

source ~/.bashrc  
를 통해 적용



```
changsu@changsu-virtual-machine: ~/workspace
1 #include <stdio.h>
2
3 void main()
4 {
5     printf("Hello World\n");
6 }
7
8
~
~
~
~
~
```

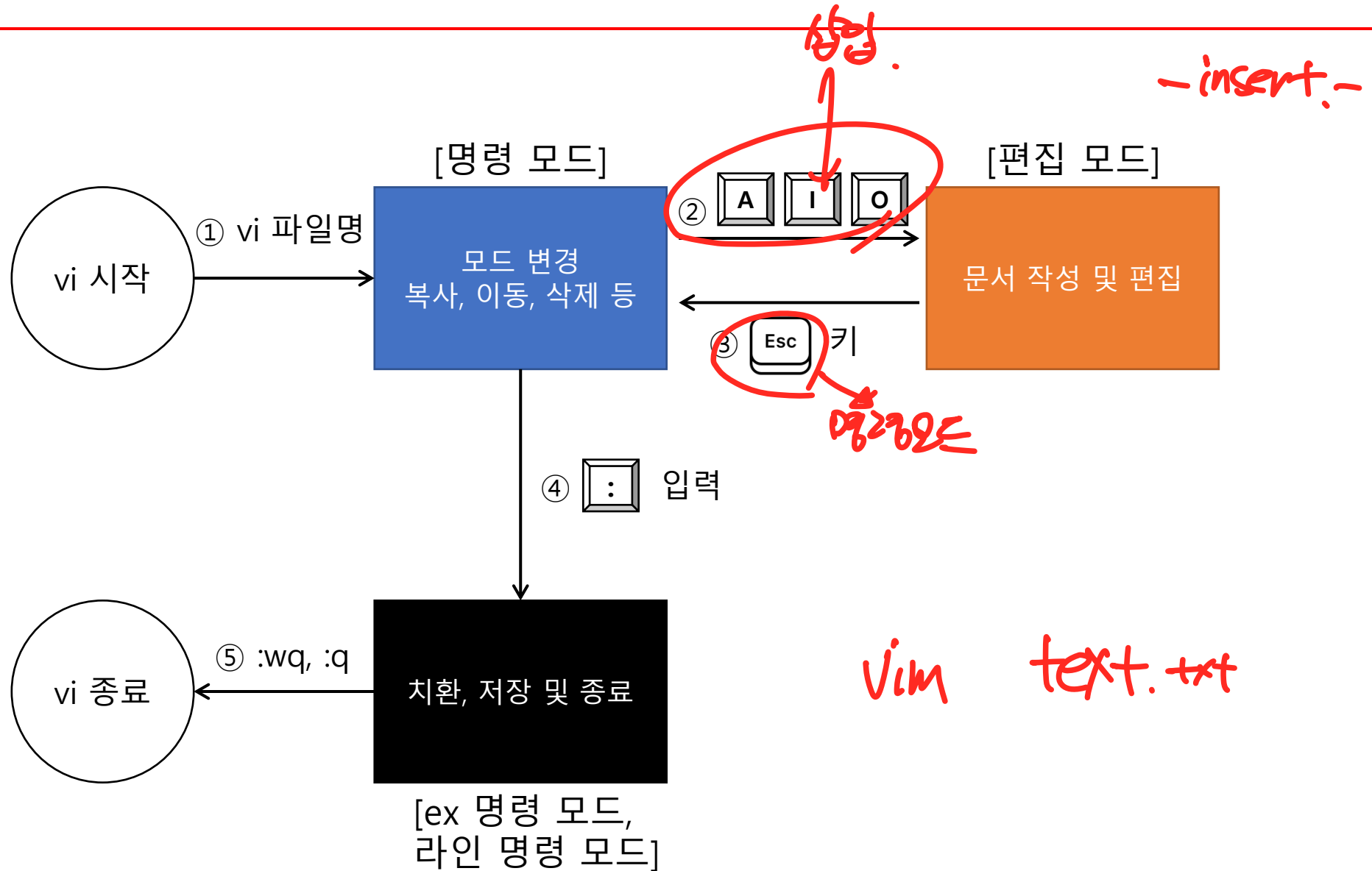


# vi 에디터 실행

## ■ vi 에디터 실행

\$ vi 파일이름

# vi 기본 동작





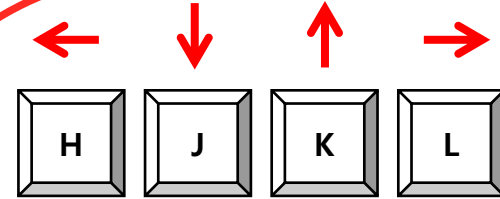
# vi 명령 모드

'a' : 현재 커서 뒤에 추가  
'o' : 다음 줄

## ■ 명령 모드: ESC

### • 커서 이동

:숫자 + Enter: 해당 라인으로 이동



### • 삭제, 복사, 붙여 넣기

한번  
재무력

| 명령어   | 의미                              |
|-------|---------------------------------|
| x     | 현재 위치에서 한 글자 삭제 (del)           |
| dw    | 단어 단위 삭제                        |
| dd    | 현재 커서 위치에서 행 삭제 (1줄 삭제)         |
| 숫자dd  | 커서 위치에서 숫자개의 라인 삭제              |
| yy    | 현재 커서 위치에서 행 복사 (1줄 복사)         |
| 숫자yy  | 현재 커서 위치에서 숫자만큼의 행을 복사 (예: 3yy) |
| p     | 복사한 내용을 현재 행 이후에 붙여 넣기          |
| P     | 복사한 내용을 현재 행 이전에 붙여 넣기          |
| :5,7d | 5~7번째 행 삭제                      |
| u     | 직전 명령 취소 (undo)                 |

# vi 입력 모드

- 입력 모드 (편집 모드)
  - i: 현재 커서의 위치부터 입력
  - a: 현재 커서의 위치 다음부터 입력
  - o: 현재 커서의 다음 줄에 입력
- ex 명령 모드 (라인 명령 모드)

| 명령어           | 의미                          |
|---------------|-----------------------------|
| :w            | 변경 사항 저장                    |
| :w 파일명        | 변경 사항을 입력한 파일명에 저장          |
| :wq           | 저장 후 종료                     |
| :q!           | 강제 종료 (변경 사항 저장하지 않음)       |
| /키워드          | 해당 문서에서 키워드를 검색             |
| n             | 검색된 키워드가 있는 경우 다음으로 이동      |
| N             | 검색된 키워드가 있는 경우 이전으로 이동      |
| :%s/str/rep/g | 파일 전체에서 str을 찾아서 rep로 전부 교체 |

# vi 단축키

version 1.1  
April 1st, 06

Esc

명령 모드

vi / vim

단축키 모음

|             |                 |             |              |             |              |             |             |            |               |            |          |           |
|-------------|-----------------|-------------|--------------|-------------|--------------|-------------|-------------|------------|---------------|------------|----------|-----------|
| ~ 대문자 전환    | ! 외부 명령         | @ 매크로 실행    | # 이전 검색      | \$ 줄 끝으로 이동 | % 일치하는 줄로 찾기 | ^ 줄의 첫 글자   | & :s 반복     | * 다음 검색    | ( 문장 시작       | ) 문장 끝     | 아래줄로 이동  | + 다음 줄    |
| · 마크로 이동    | 1               | 2           | 3            | 4           | 5            | 6           | 7           | 8          | 9             | 0 줄의 처음    | - 이전 줄   | = 자동 들여쓰기 |
| Q 실행 모드     | W 다음 WORD       | E 끝 WORD    | R 수정 모드      | T 뒤로 검색     | Y 줄단위 복사     | U 줄 단위 실행취소 | I 줄 시작에서 삽입 | O 행 위에 삽입  | P 커서 이전에 붙여넣기 | { 문단 시작    | }        | 문단 끝      |
| q 매크로 기록    | w 다음 단어         | e 단어 끝      | r 한 문자 교체    | t 한 문자 검색   | y 복사         | u 실행취소      | i 편집 모드     | o 행 아래에 삽입 | p 커서 이후에 붙여넣기 | [ 기타       | ]        | 기타        |
| A 줄 끝에 덧붙이기 | S 삭제 후 편집 모드    | D 줄 끝까지 삭제  | F 뒤로 검색      | G 파일 끝으로 이동 | H 화면 상단      | J 줄 합치기     | K 다음 말      | L 화면 하단    | : 명령줄         | ex 레지스터 지정 | " 열 이동   | .         |
| a 덧붙이기      | s 단어 삭제 후 편집 모드 | d 삭제        | f 한 문자 찾기    | g 확장 명령     | h ←          | j ↓         | k ↑         | l →        | ; 명령 반복       | /T/t/F/f   | ' 마크로 이동 | \ 사용 안함   |
| Z 종료        | X 백스페이스 바꾸기     | C 줄 끝까지 바꾸기 | V 줄단위 비주얼 모드 | B 이전 WORD   | N 이전 (찾기)    | M 화면 가운데    | < 내어쓰기      | > 들여쓰기     | ? 찾기 (뒤로)     | .          | .        | .         |
| Z 확장명       | x 글자 삭제         | c 바꾸기       | v 비주얼 모드     | b 이전 단어     | n 다음 (찾기)    | m 마크 설정     | , 역순 검색     | /T/t/F/f   | .             | .          | .        | .         |

동작

명령

연산자

확장

커서를 이동하거나, 연산자가 동작할 범위를 지정합니다.

바로 동작하는 명령, 빨간색은 편집 모드로 변경됩니다.

이동 관련 문자(숫자나 커서 이동)와 함께 사용하여야 하며, 커서의 위치부터 목적지까지 연산합니다.

특별한 키 합수로, 추가적인 키 입력이 필요합니다.

입력 후 (숫자를 제외한 .으로 끝낼 수 있는) 글자를 입력하여야 합니다.

words: 구분자로 공백, 특수기호 모두 사용  
WORDS: 구분자로 공백 문자만 사용

words: `quux(foo, baz, baz)`  
WORDS: `quux(foo, baz, baz)`

주요 명령행 명령 ('ex'):

:w (저장), :q (종료), :q! (저장하지 않고 종료)  
:e f (파일 f 열기),  
:%s/x/y/g (파일 전체에서 'x'를 'y'로 교체),  
:h (vim 도움말), :new (새 파일)

그외 중요한 명령들:

CTRL-R: 재실행 (vim),  
CTRL-F/-B: 페이지 위로/아래로,  
CTRL-E/-Y: 줄 스크롤 위로/아래로,  
CTRL-V: 블록-비주얼 모드 (vim 전용)

비주얼 모드:

커서를 움직여 지정된 범위에 연산자를 적용합니다. (vim 전용)

참고:

(1) 복사/붙여넣기/지우기 명령어를 사용하기 전에 "x"를 입력하여 레지스터(클립보드)를 지정하세요. (x는 a에서 z 또는 \*을 사용할 수 있음) (예: "ay\$를 입력하면 현재 커서에서 라인 끝까지의 내용을 레지스터 'a'에 저장합니다.)

(2) 어떤 명령을 입력하기 전에 횡수를 지정하면, 횡수만큼 반복하게 됩니다. (예: 2p, d2w, 5i, d4j)

(3) 연속으로 입력하는 명령은 현재의 라인에 반영됩니다. 예시: dd(현재 라인 지우기), >>(들여쓰기)

(4) ZZ는 저장후 종료, ZQ는 저장하지 않고 종료.

(5) zt : 커서가 위치한 곳을 제일위로 올리기, zb : 바닥으로, zz : 가운데로

(6) gg : 파일의 처음으로(Vim 전용), gf : 커서가 위치한 곳의 파일 열기(Vim 전용)

# 리눅스 메뉴얼 페이지 (man)

- 사용 방법
  - `$ man 명령어 이름`
- 메뉴얼 보기
  - `$ man man`

| 단락 | 설명                                 |
|----|------------------------------------|
| 1  | 일반 명령어, shell 명령어                  |
| 2  | 시스템 호출                             |
| 3  | C 표준 라이브러리 함수들                     |
| 4  | 특수파일 (/dev의 장치 파일)과 드라이버           |
| 5  | 파일 포맷과 conventions (/etc/passwd 등) |
| 6  | 게임                                 |
| 7  | 기타                                 |
| 8  | 시스템 관리 명령어와 데몬                     |

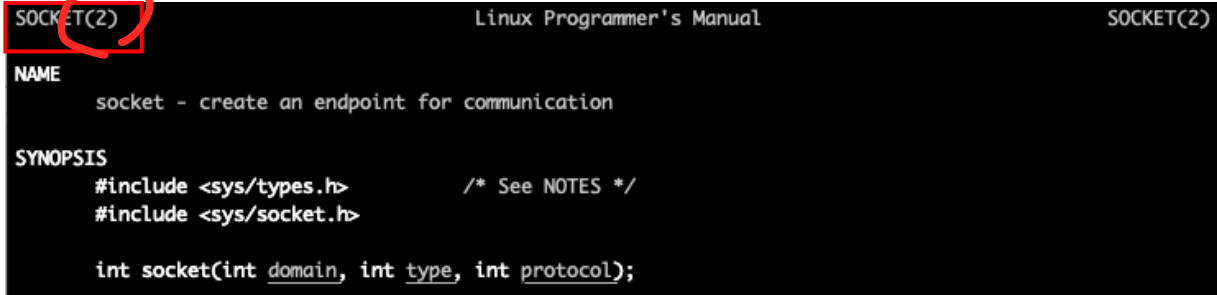
라이브러리 printf의 메뉴얼 확인

```
$ man 3 printf
```

# man 예제

## ■ socket() 함수

- 시스템 호출(2)
- `$man socket`



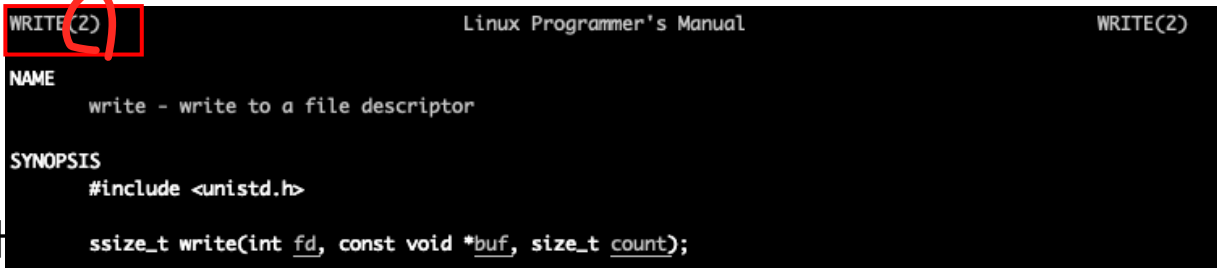
```
SOCKET(2)                                Linux Programmer's Manual                                SOCKET(2)
NAME
    socket - create an endpoint for communication

SYNOPSIS
    #include <sys/types.h>          /* See NOTES */
    #include <sys/socket.h>

    int socket(int domain, int type, int protocol);
```

## ■ read(), write() 함수

- 시스템 호출(2)
- `$ man 2 write`



```
WRITE(2)                                Linux Programmer's Manual                                WRITE(2)
NAME
    write - write to a file descriptor

SYNOPSIS
    #include <unistd.h>

    ssize_t write(int fd, const void *buf, size_t count);
```

## ■ printf

- 명령어(1), C 라이브러리(3) 모두 존재
- `$ man 3 printf`



```
PRINTF(3)                                Linux Programmer's Manual                                PRINTF(3)
NAME
    printf, fprintf, sprintf, snprintf, vprintf, fprintf, vsprintf, vsnprintf - formatted output
    conversion

SYNOPSIS
    #include <stdio.h>

    int printf(const char *format, ...);
    int fprintf(FILE *stream, const char *format, ...);
    int sprintf(char *str, const char *format, ...);
    int snprintf(char *str, size_t size, const char *format, ...);
```

# Questions?

LMS Q&A 게시판에 올려주세요.