2. Directory and File

Hyunchan, Park

http://oslab.jbnu.ac.kr

Division of Computer Science and Engineering

Jeonbuk National University

학습 내용

- 디렉토리 구조와 관련 명령어 실습

- 파일과 관련 명령어 실습

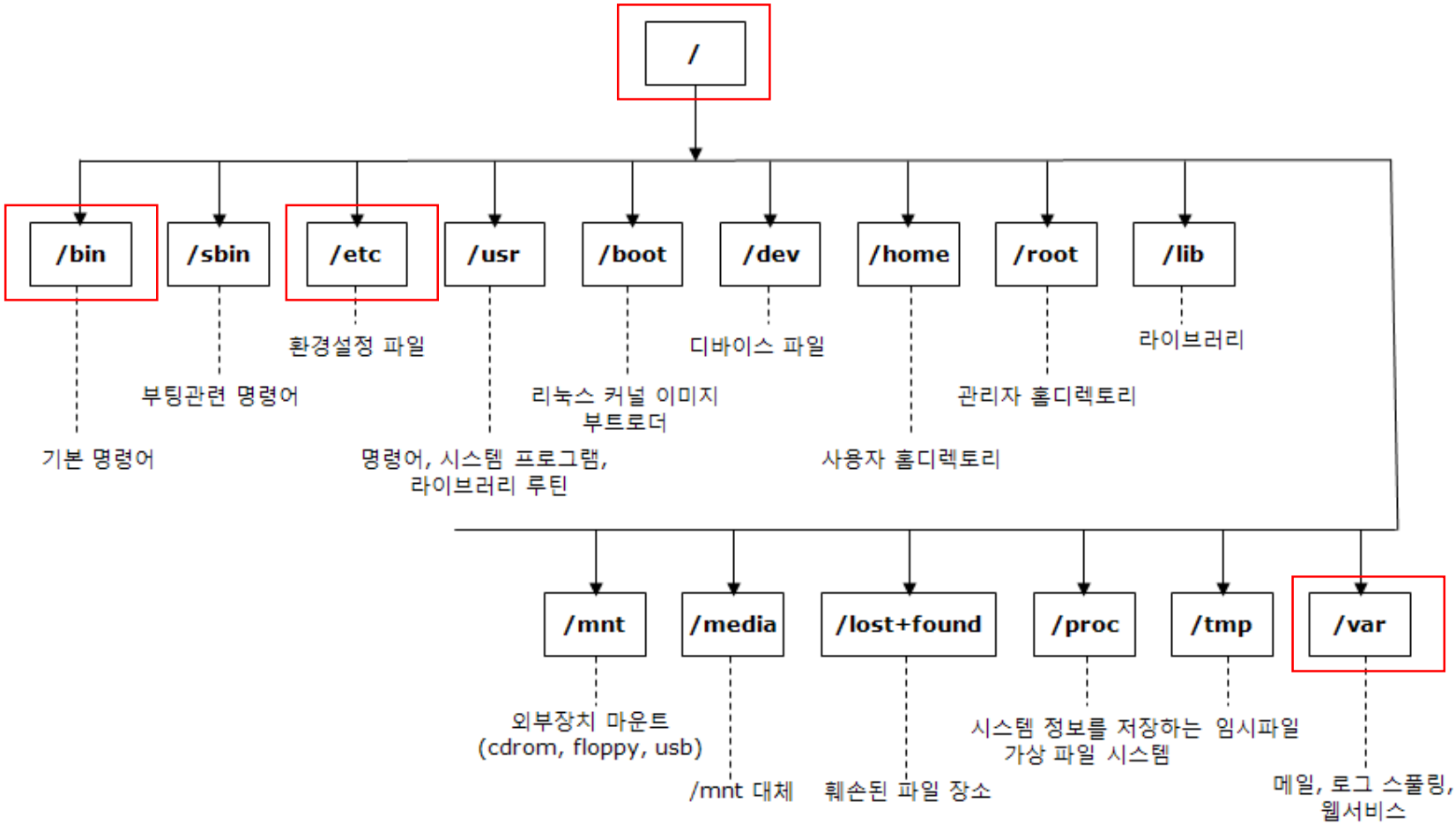
개인 과제 2: 실습 및 C언어 복습

- 실습 과제
  - 실습 내용에서 다루는 명령어를 모두 입력하고, 그 결과를 확인할 것
    - 동영상에서 수행한 내용
  - 제출 방법
    - Old LMS, 개인 과제 2
    - Xshell 로그 파일 1개 제출
    - 파일 명: 학번.txt
- C언어 복습 과제
  - JOTA: hw2-1,hw2-2 문제 수행
- 제출 기한
  - 9/22 (화) 23:59 (지각 감점: 5%p / 12H, 1주 이후 제출 불가)

Directory

디렉터리 계층구조

- 리눅스의 디렉터리는 루트로부터 시작하여 트리 형태의 계층구조를 이룬다.



디렉터리 계층구조: Root (“/”) directory

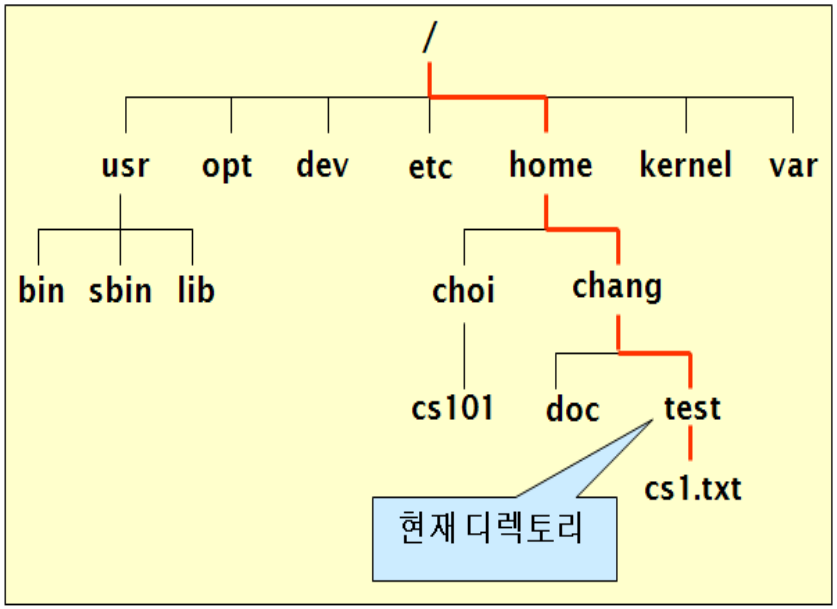
- 디렉토리 계층 구조의 뿌리
  - 모든 정보는 Root 에서부터 단일한 트리 구조로 구성됨
- Windows의 경우
  - 각각의 저장 장치 파티션마다 드라이브 문자 (C, D 드라이브) 지정
  - 각 드라이브마다 루트가 존재하고 디렉토리 구조가 존재하는 것
- Root 를 관리하는 파일 시스템을 루트 파일 시스템이라고 함

```
ubuntu@unix:~$ mount | grep /dev/vda1
/dev/vda1 on / type ext4 (rw,relatime)
```

  - 예) Ext4 라는 파일 시스템이 현재 루트를 관리하고 있음
  - /dev/vda1 이라는 저장 장치에 루트 파일 시스템 내부의 정보가 저장됨
- 리눅스에 새로운 저장 장치가 추가되면?
  - 루트 파일 시스템 하의 어떤 디렉토리에 해당 장치를 마운트시킬 수 있음
  - 예) /home/hcpark/new_ssd 라는 디렉토리를 만들고, 새 장치 Mount 를 수행하면 해당 디렉토리 밑에 저장되는 정보는 루트 파일 시스템이 아닌 새로운 장치에 저장됨

홈 디렉터리

- 홈 디렉터리(home directory)
  - 각 사용자마다 별도의 홈 디렉터리가 있음
  - 사용자가 로그인하면 홈 디렉터리에서 작업을 시작함



~ : 홈 디렉터리
. : 현재 디렉터리
.. : 부모 디렉터리

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

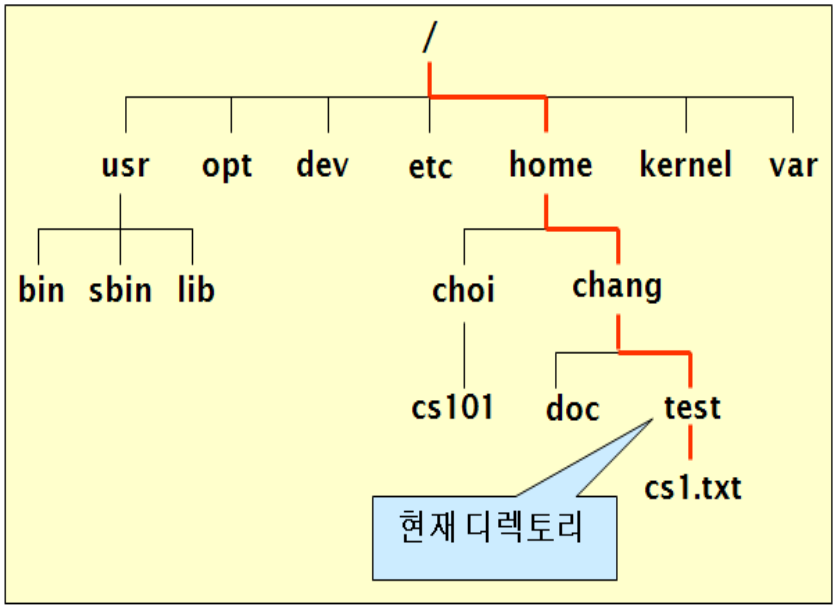
cs1.txt의 상대 경로명
cs1.txt

기타 자주 접근하는 디렉토리

- /bin
  - Binaries: System-wide (global) 하게 사용되는 기본적인 명령어에 대한 실행 파일들
- /sbin
  - System binaries: 시스템 관리를 위한 명령어의 실행 파일들
- /lib
  - Libraries: System-wide (global) 하게 사용되는 라이브러리 파일들
- /usr
  - 각 사용자들이 설치한 프로그램들의 실행파일과 라이브러리, 소스, 매뉴얼 등
  - /usr/local: 새로운 프로그램들이 설치됨 (windows에서 c:\program files\ 와 유사함)
- /etc
  - Configurations: 각종 환경 설정 파일들 (네트워크, 서비스, 사용자, 암호 등등)
- /var
  - Variable data: 시스템 동작 중에 계속해서 변화하는 파일들 (로그, 웹데이터, lock)

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



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

cs1.txt의 상대 경로명
cs1.txt

현재 작업 디렉터리 출력: pwd(print working directory)

- 사용법

```
$ pwd
```

현재 작업 디렉터리의 절대 경로명을 출력한다.

- 현재 작업 디렉터리(current working directory)
  - 현재 작업 중인 디렉터리
  - 로그인 하면 홈 디렉터리에서부터 작업이 시작된다.

- 예

```
$ pwd
/home/chang/Desktop
$ cd ~
$ pwd
/home/chang
```

디렉터리 이동: cd (change directory)

- 사용법

```
$ cd [디렉터리]
```

현재 작업 디렉터리를 지정된 디렉터리로 이동한다.

디렉터리를 지정하지 않으면 홈 디렉터리로 이동한다.

- 예

```
$ cd
$ cd ~
$ cd Desktop
$ pwd
/home/chang/Desktop
$ cd ..
```

명령어의 경로 확인: which

- 사용법

```
$ which 명령어
```

명령어의 절대경로를 보여준다.

- 예

```
$ which ls
/bin/ls
$ which pwd
/usr/pwd
$ which passwd
/usr/passwd
```

디렉터리 리스트: ls(list)

- 사용법

> $ ls(혹은 dir) [-aslFR] 디렉터리* 파일*
>
> 지정된 디렉터리의 내용을 리스트 한다. 디렉터리를 지정하지 않으면 현재 디렉터리 내용을 리스트 한다. 또한 파일을 지정하면 해당 파일만을 리스트 한다.

- 예

```
$ ls /
bin dev home lib64 mnt proc run srv tmp var
boot etc lib media opt root sbin sys usr
$ ls ~
Desktop Downloads Pictures Templates pl 다운로드
Documents Music Public Videos linux tmp 사진
$ cd Desktop
$ ls
cs1.txt
```

ls 명령어 옵션

- 주요 옵션

옵션	기능
-a	숨겨진 파일을 포함하여 모든 파일을 리스트한다.
-s	파일의 크기를 K 바이트 단위로 출력한다.
-l	파일의 상세 정보를 출력한다.
-F	파일의 종류를 표시하여 출력한다.
-R	모든 하위 디렉터리들을 리스트한다.
-t	최근 수정된 시간 순서로 정렬하여 출력한다.

ls 명령어 옵션

- **ls -s**
  - -s(size) 옵션
  - 디렉터리 내에 있는 모든 파일의 크기를 K 바이트 단위로 출력

```
$ ls -s
총 4
4 cs1.txt
```

- **ls -a**
  - -a(all) 옵션
  - 숨겨진 파일들을 포함하여 모든 파일과 디렉터리를 리스트
  - "."은 현재 디렉터리, ".."은 부모 디렉터리

```
$ ls -a
. .. cs1.txt
```

ls 명령어 옵션

- **ls -l**
  - -l(long) 옵션
  - 파일 속성(file attribute) 출력
    - 파일 이름, 파일 종류, 접근권한, 소유자, 크기, 수정 시간 등

- **ls –asl**

```
$ ls -asl
총 8
0 drwxr-xr-x 2 chang cs 20 4월 16일 13:37 .
4 drwx------ 3 chang cs 4096 4월 16일 13:37 ..
4 -rw-r--r-- 1 chang cs 2088 4월 16일 13:37 cs1.txt
```

ls 명령어 옵션

- **ls –asl**

```
$ ls -asl
총 8
0 drwxr-xr-x 2 chang cs 20 4월 16일 13:37 .
4 drwx------ 3 chang cs 4096 4월 16일 13:37 ..
4 -rw-r--r-- 1 chang cs 2088 4월 16일 13:37 cs1.txt
```

ls 명령어 옵션

- **ls –F**
  - 기호로 파일의 종류를 표시

  *: 실행파일, /: 디렉터리, @:심볼릭 링크

- 예

```
$ ls -F /
bin@ dev/ home/ lib64@ mnt/ proc/ run/ srv/ tmp/ var/
boot/ etc/ lib@ media/ opt/ root/ sbin@ sys/ usr/
```

ls 명령어 옵션

- **ls –R**
  - -R(Recursive) 옵션
  - 모든 하위 디렉터리 내용을 리스트 한다.

- 예

```
$ ls -R
$ ls -R /
```

디렉터리 생성: mkdir(make directory)

- 사용법

```
$ mkdir [-p] 디렉터리+
```

디렉터리(들)을 새로 만든다.

- 예

```
$ cd ~      // 홈 디렉터리로 이동
$ mkdir test
$ mkdir test temp
$ ls -l
drwxrwxr-x. 2 chang chang 6 5월 12 10:12 temp
drwxrwxr-x. 2 chang chang 6 5월 12 10:12 test
```

디렉터리 생성: mkdir

- 중간 디렉터리 자동 생성 옵션 -p
  - 필요한 경우에 중간 디렉터리를 자동으로 만들어 준다.

- 예 : ~/dest 디렉터리가 없는 경우

```
$ mkdir ~/dest/dir1
mkdir: '/home/chang/dest/dir1' 디렉터리를 만들 수 없습니다: 그런 파일이나 디렉터리가 없습니다
$ mkdir -p ~/dest/dir1
```

디렉터리 삭제 : rmdir(remove directory)

- 사용법

$ rmdir 디렉터리+ 디렉터리(들)을 삭제한다.

  - 주의: 빈 디렉토리만 삭제할 수 있다.

- 예

```
$ rmdir test
rmdir: failed to remove ‘test’: 디렉터리가 비어있지 않음
```

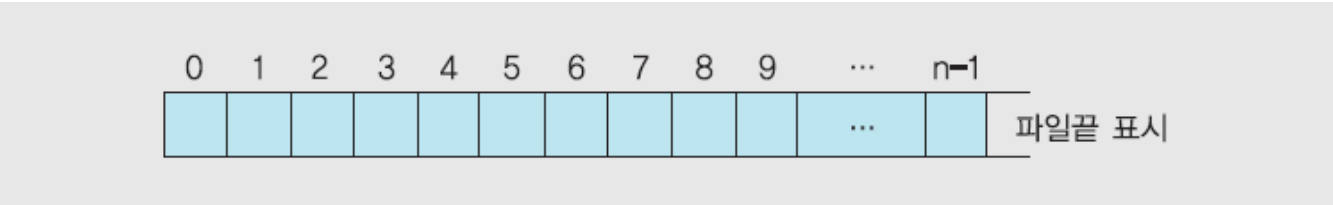
File

- Basic, Stat, and Print out
- Copy, Move, and Remove
- Link

파일

- 프로그래밍에서 파일은 왜 필요할까?
  - 변수에 저장된 정보들은 실행이 끝나면 모두 사라진다.
  - 정보를 영속적으로 저장하기 위해서는 파일에 저장해야 한다.

- 유닉스 파일
  - 모든 데이터를 연속된 바이트 형태로 저장한다.
    -



파일의 종류

- 일반 파일(ordinary file)
  - 데이터를 가지고 있으면서 디스크에 저장된다.
  - 텍스트 파일, 이진 파일 (binary file)

- 디렉터리(directory) 또는 폴더(folder)
  - 파일들을 계층적으로 조직화하는 데 사용되는 일종의 특수 파일
  - 디렉터리 내에 파일이나 서브디렉토리들이 존재한다.

- 장치 파일(device special file)
  - 물리적인 장치에 대한 내부적인 표현
  - 키보드(stdin), 모니터(stdout), 프린터 등도 파일처럼 사용

- 심볼릭 링크 파일
  - 어떤 파일을 가리키는 또 하나의 경로명을 저장하는 파일

파일 종류

- 리눅스에서 지원하는 파일 종류

파일 종류	표시	설명
일반 파일	-	데이터를 갖고 있는 텍스트 파일 또는 이진 파일
디렉터리 파일	d	디렉터리 내의 파일들의 이름들과 파일 정보를 관리하는 파일
문자 장치 파일	c	문자 단위로 데이터를 전송하는 장치를 나타내는 파일
블록 장치 파일	b	블록 단위로 데이터를 전송하는 장치를 나타내는 파일
FIFO 파일	p	프로세스 간 통신에 사용되는 이름 있는 파이프
소켓	s	네트워크를 통한 프로세스 간 통신에 사용되는 파일
심볼릭 링크	l	다른 파일을 가리키는 포인터와 같은 역할을 하는 파일

File 명령어

- 사용법

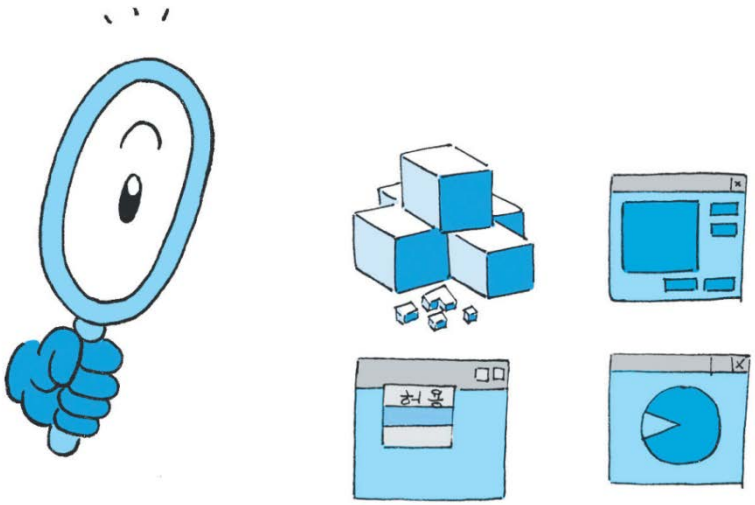
```
$ file [옵션] 파일

Determine type of FILEs.
```

- 예

```
unix201512345@unix:~$ file /proc/cpuinfo
/proc/cpuinfo: empty
unix201512345@unix:~$ file .cache/
.cache/: directory
unix201512345@unix:~$ file newfile
newfile: ASCII text
unix201512345@unix:~$ file /usr/bin/file
/usr/bin/file: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamical
ly linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=32715f59ea258e
8fdf0dd8763fc501f958b0c4d6, for GNU/Linux 3.2.0, stripped
```

파일 상태(file status)

- 파일 상태
  - 파일에 대한 모든 정보
  - 블록수, 파일 타입, 접근권한,
  - 링크수, 파일 소유자의 사용자 ID,
  - 그룹 ID, 파일 크기, 최종 수정 시간, 등
  - 메타데이터 (metadata)
- 예)



```
unix201512345@unix:~$ ls -l newfile
-rw-rw-r-- 1 unix201512345 unix201512345 14 Sep  8 08:27 newfile
```

접근권한 링크수 소유자ID 소유그룹ID 파일 크기 최종 수정 시간 파일 이름

파일 속성(file attribute)

- 파일 크기, 종류, 접근권한, 링크 수, 소유자 및 그룹, 수정 시간

파일 속성	의미
파일 크기	파일의 크기(K 바이트 단위)
파일 종류	일반 파일(-), 디렉터리(d), 링크(l), 파이프(p), 소켓(s), 디바이스(b 혹은 c) 등의 파일 종류를 나타낸다.
접근권한	파일에 대한 소유자, 그룹, 기타 사용자의 읽기(r)/쓰기(w)/실행(x) 권한
하드 링크 수	파일에 대한 하드 링크 개수
소유자 및 그룹	파일의 소유자 ID 및 소유자가 속한 그룹
파일 크기	파일의 크기(바이트 단위)
최종 수정 시간	파일을 생성 혹은 최후로 수정한 시간

stat 명령어

- 사용법

```
$ stat [옵션] 파일
```

파일의 자세한 상태 정보를 출력한다.

- 예

```
unix201512345@unix:~$ stat newfile
  File: newfile
  Size: 14              Blocks: 8          IO Block: 4096   regular file
Device: fc01h/64513d    Inode: 270312      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1001/unix201512345)   Gid: ( 1001/unix201512345)
Access: 2020-09-08 08:27:04.169145344 +0000
Modify: 2020-09-08 08:27:00.185212452 +0000
Change: 2020-09-08 08:27:00.185212452 +0000
 Birth: -
```

Blocks: 실제 저장 장치 내에서 차지하는 블록 개수
IO Block: 1개 블록의 크기 (바이트). 이 단위로 장치 입출력이 수행됨
I-node: Index node. 운영체제에서 해당 파일을 관리하기 위해 부여한 번호
Links: 해당 I-node와 연결된 파일의 개수

파일 내용 출력

- 파일 내용 출력과 관련된 다음 명령어들
  - cat, more, head, tail, wc 등
- 명령어 테스트를 위해 파일 하나를 카피해서 사용

```
cd ~
cp /etc/services temp.txt
ls -al
```

```
unix201512345@unix:~$ cp /etc/services ./temp.text
unix201512345@unix:~$ ls -al
total 64
drwxr-xr-x   5 unix201512345 unix201512345  4096 Sep 10 01:37 .
drwxr-xr-x 108 root          root           4096 Sep  7 14:40 ..
-rw-------   1 unix201512345 unix201512345   200 Sep  9 04:37 .Xauthority
-rw-------   1 unix201512345 unix201512345   948 Sep  9 04:52 .bash_history
-rw-r--r--   1 unix201512345 unix201512345   220 Sep  7 06:57 .bash_logout
-rw-r--r--   1 unix201512345 unix201512345  3771 Sep  7 06:57 .bashrc
drwx------   2 unix201512345 unix201512345  4096 Sep  7 08:26 .cache
drwx------   3 unix201512345 unix201512345  4096 Sep  8 08:09 .config
drwxrwxr-x   3 unix201512345 unix201512345  4096 Sep  7 15:17 .local
-rw-r--r--   1 unix201512345 unix201512345   807 Sep  7 06:57 .profile
-rw-------   1 unix201512345 unix201512345  2044 Sep 10 01:09 .viminfo
-rw-rw-r--   1 unix201512345 unix201512345    20 Sep  9 04:52 newfile
-rw-r--r--   1 unix201512345 unix201512345 14464 Sep 10 01:37 temp.text
unix201512345@unix:~$
```

파일 내용 보기: cat

- 사용법

$ cat [-n] 파일

파일(들)의 내용을 그대로 화면에 출력한다. 파일을 지정하지 않으면 표준입력 내용을 그대로 화면에 출력한다.

옵션: -n, --number number all output lines

- 예

```
$ cat cs1.txt
Unix is a multitasking, multi-user computer operating system originally
developed in 1969 by a group of AT&T employees at Bell Labs, including
Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy,
and Joe Ossanna.
...
```

파일 내용 보기: cat

- 예

```
$ cat –n cs1.txt
1 Unix is a multitasking, multi-user computer operating system originally
2 developed in 1969 by a group of AT&T employees at Bell Labs, including
3 Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy,
4 and Joe Ossanna.
...

$ cat                  // 지정 파일 없는 경우, 내용 입력하여 새로운 파일 생성 가능
Hello World !
Hello World !
Bye!
Bye!
^D
```

페이지 단위로 파일 내용 보기: more

- 사용법

> $ more 파일
>
> 파일(들)의 내용을 페이지 단위로 화면에 출력한다.

- 예

```
$ more cs1.txt
Unix is a multitasking, multi-user computer operating system originally
developed in 1969 by a group of AT&T employees at Bell Labs, including
Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy,
and Joe Ossanna.
...
During the late 1970s and early 1980s, the influence of Unix in academic
circles led to large-scale adoption of Unix(particularly of the BSD variant,
--계속--(59%)
```

파일 앞부분보기: head

- 사용법

> $ head [-*n*] 파일
>
> 파일(들)의 앞부분을 화면에 출력한다. 파일을 지정하지 않으면 표준입력 내용을 대상으로 한다.

- 예

```
$ head -5 cs1.txt
Unix is a multitasking, multi-user computer operating system originally
developed in 1969 by a group of AT&T employees at Bell Labs, including
Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy,
and Joe Ossanna.
```

파일 뒷부분보기: tail

- 사용법

  $ tail [-*n*] 파일

  파일(들)의 뒷 10개 라인을 화면에 출력한다. 파일을 지정하지 않으면 표준입력 내용을 대상으로 한다.

  옵션: -f, --follow[={name|descriptor}] output appended data as the file grows;

- 예

```
$ tail cs1.txt
Linux, which is used to power data centers, desktops, mobile phones,
and embedded devices such as routers, set-top boxes or e-book readers.
Today, in addition to certified Unix systems such as those already
mentioned, Unix-like operating systems such as MINIX, Linux, Android,
and BSD descendants (FreeBSD, NetBSD, OpenBSD, and DragonFly BSD) are
commonly encountered.
The term traditional Unix may be used to describe a Unix or
an operating system that has the characteristics of either Version 7
Unix or UNIX System V.
```

단어 세기: wc (word count)

- 사용법

  $ wc [-lwc] 파일

  파일에 저장된 줄(l), 단어(w), 문자(c)의 개수를 세서 출력한다.

  파일을 지정하지 않으면 표준입력 내용을 대상으로 한다.

- 예

```
$ wc cs1.txt
38 318 2088 cs1.txt
$ wc -l cs1.txt
38 cs1.txt
$ wc -w cs1.txt
318 cs1.txt
$ wc -c cs1.txt
2088 cs1.txt
```

```
unix201512345@unix:~$ wc --help
Usage: wc [OPTION]... [FILE]...
  or:  wc [OPTION]... --files0-from=F
Print newline, word, and byte counts for each FILE, and a total line if
more than one FILE is specified.  A word is a non-zero-length sequence of
characters delimited by white space.

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

The options below may be used to select which counts are printed, always in
the following order: newline, word, character, byte, maximum line length.
  -c, --bytes            print the byte counts
  -m, --chars            print the character counts
  -l, --lines            print the newline counts
      --files0-from=F    read input from the files specified by
                           NUL-terminated names in file F;
                           If F is - then read names from standard input
  -L, --max-line-length  print the maximum display width
  -w, --words            print the word counts
      --help     display this help and exit
      --version  output version information and exit

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Report wc translation bugs to <https://translationproject.org/team/>
Full documentation at: <https://www.gnu.org/software/coreutils/wc>
or available locally via: info '(coreutils) wc invocation'
```

File

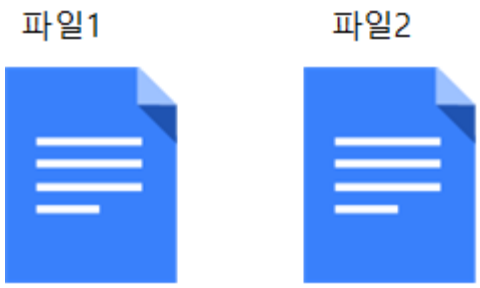
- Basic, Stat, and Print out
- Copy, Move, and Remove
- Link

파일 복사: cp(copy)

- 사용법

```
$ cp [-i] 파일1 파일2
```

파일1을 파일2에 복사한다. -i는 대화형 옵션이다.



- 예

```
$ cp cs1.txt cs2.txt
$ ls -l cs1.txt cs2.txt
-rw-r--r-- 1 chang cs 2088 4월 16일 13:37 cs1.txt
-rw-r--r-- 1 chang cs 2088 4월 16일 13:45 cs2.txt
$ cp /etc/hosts hostnames
```

파일 복사: cp(copy)

- 대화형 옵션: cp -i
  - 복사 대상 파일과 이름이 같은 파일이 이미 존재하면 덮어쓰기(overwrite) !
  - 보다 안전한 사용법: 대화형 –i(interactive) 옵션을 사용

- 예

```
$ cp –i cs1.txt cs2.txt
cp: overwrite ‘cs2.txt’? n
```

파일 복사: cp(copy)

- 파일을 디렉터리로 복사

```
$ cp 파일 디렉터리
```

파일을 지정된 디렉터리에 복사한다.

```
$ cp 파일1 ... 파일n 디렉터리
```

여러 개의 파일들을 지정된 디렉터리에 모두 복사한다.

- 예

```
$ cp cs1.txt /tmp
$ ls -l /tmp/cs1.txt
-rw-r--r-- 1 chang cs 2088 4월 16일 14:31 /tmp/cs1.txt
$ cp cs1.txt cs2.txt /tmp
```

파일 복사: cp(copy)

- 디렉터리 전체 복사 : cp –r

```
$ cp [-r] 디렉터리1 디렉터리2
```

r은 리커전 옵션으로 디렉터리1 전체를 디렉터리2에 복사한다.

  - 하위 디렉터리를 포함한 디렉터리 전체를 복사

- 예

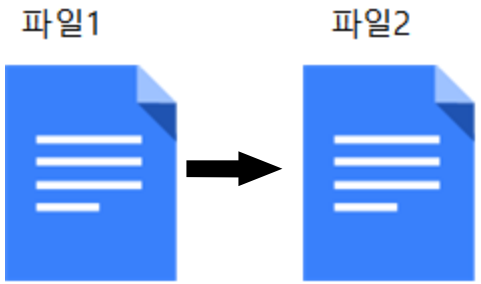
```
$ cp –r test temp
```

파일 이동: mv(move)

- 사용법

$ mv [-i] 파일1 파일2

파일1의 이름을 파일2로 변경한다. -i는 대화형 옵션이다.



- 예

```
$ 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
```

파일 이동: mv(move)

- 대화형 옵션: mv -i
  - 이동 대상 파일과 이름이 같은 파일이 이미 존재하면 덮어쓰기 (overwrite)
  - 보다 안전한 사용법: 대화형 –i(interactive) 옵션을 사용

- 예

```
$ mv –i cs1.txt cs3.txt
mv: overwrite ‘cs3.txt’? n
```

파일 이동: mv(move)

- 파일을 디렉터리로 이동

```
$ mv 파일 디렉터리
```

파일을 지정된 디렉터리로 이동한다.

```
$ mv 파일1 ... 파일n 디렉터리
```

여러 개의 파일들을 지정된 디렉터리로 모두 이동한다.

- 예

```
$ mv cs3.txt /tmp
$ ls -l /tmp/cs3.txt
-rw-r--r-- 1 chang cs 2088 4월 16일 13:56 /tmp/cs3.txt
$ mv cs1.txt cs3.txt /tmp
```

파일 이동: mv(move)

- 두 인자가 모두 디렉터리인 경우, 디렉토리 이름이 변경됨

```
$ mv 디렉터리1 디렉터리2
```

디렉터리1을 지정된 디렉터리2로 이름을 변경한다.

- 예

```
$ mkdir temp
$ mv temp tmp
```

파일 삭제: rm(remove)

- 사용법

```
$ rm [-i] 파일+
```

파일(들)을 삭제한다. -i는 대화형 옵션이다.

- 예

```
$ rm cs1.txt
$ rm cs1.txt cs3.txt
```

- 대화형 옵션 : rm -i

```
$ rm –i cs1.txt
rm: remove 'cs1.txt'? n
```

디렉터리 전체 삭제

- 디렉터리 전체 삭제: rm –r (디렉토리 포함, rmdir 대신 사용 가능)

```
$ rm [-ri] 디렉터리
```

-r은 리커전 옵션으로 디렉터리 아래의 모든 것을 삭제한다. -i는 대화형 옵션

- 예

```
$ rm test
rm: cannot remove 'test': 디렉터리입니다
$ rmdir test
rmdir: failed to remove 'test': 디렉터리가 비어있지 않음
$ rm –ri test
rm: descend into directory 'test'? y
rm: remove regular file 'test/cs3.txt'? y
Rm: remove regular file ‘test/cs1.txt’? y
rm: remove directory 'test'? y
```

File

- Basic, Stat, and Print out
- Copy, Move, and Remove
- Link

링크

- 링크
  - 기존 파일에 대한 또 하나의 새로운 이름

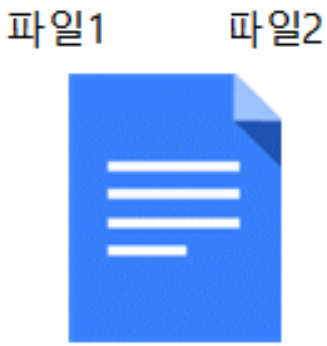
- 사용법

```
$ ln [-s] 파일1 파일2
```

파일1에 대한 새로운 이름(링크)로 파일2를 만들어 준다. -s 옵션은 심볼릭 링크

```
$ ln [-s] 파일1 디렉터리
```

파일1에 대한 링크를 지정된 디렉터리에 같은 이름으로 만들어 준다.



하드 링크(hard link)

- 하드 링크
  - 기존 파일에 대한 새로운 이름이라고 생각할 수 있다.
  - 실제로 기존 파일을 대표하는 i-노드를 가리켜 구현한다.

- 예

```
$ ln hello.txt hi.txt
$ ls -l
-rw------- 2 chang cs 15 11월 7일 15:31 hello.txt
-rw------- 2 chang cs 15 11월 7일 15:31 hi.txt
```

- 질문
  - 이 중에 한 파일의 내용을 수정하면 어떻게 될까?
  - 이 둘 중에 한 파일을 삭제하면 어떻게 될까?

심볼릭 링크(symbolic link)

- 심볼릭 링크
  - 다른 파일을 가리키고 있는 별도의 파일이다.
  - 실제 파일의 경로명을 저장하고 있는 일종의 특수 파일이다.
  - 이 경로명이 다른 파일에 대한 간접적인 포인터 역할을 한다.

- 예

```
$ ln -s hello.txt hi.txt
$ ls -l
-rw------- 1 chang cs 15 11월 7일 15:31 hello.txt
lrwxrwxrwx 1 chang cs 9 1월 24일 12:56 hi.txt -> hello.txt

$ ln -s /usr/bin/gcc cc
$ ls -l cc
lrwxrwxrwx. 1 chang chang 12 7월 21 20:09 cc -> /usr/bin/gcc
```