

UNIX/LINUX - 2nd Lab

1. Permission

- **Permission**

- 모든 UNIX 파일과 디렉토리는 3가지 타입의 permission을 가진다. (read, write, execute)
- Permission은 owner, group, others에 대해 각각 따로 설정된다.

<u>Permission</u>	<u>File</u>	<u>Directory</u>
read	User can look at the contents of the file	User can list the files in the directory
write	User can modify the contents of the file	User can create new files and remove existing files in the directory
execute	User can use the filename as a UNIX command	User can change into the directory, but cannot list the files unless (s)he has read permission. User can read files if (s)he has read permission on them.

➤ **The permission bits**

- the topmost three bits : control access for the **owner**
- the second three bits : control access for the **group**
- the last three bits : control access for **others**

```
$ ls -l  
drwxr-xr-x 2 comp-ta comp-ta 4096 2007-12-17 05:40 bash_test
```

➤ **Examples**

drwxr-xr-x : directory, the owner of the directory may read, write, execute the directory. Users in the group and all other users may read and execute the directory.

rw----- :

rw-rw-rwx :

- **chmod : 파일의 접근 권한을 바꿈**

- **Permission encoding for chmod**

r	Read	4
w	Write	2
x	Execute	1
-	No permission	0

Permissions	Octal	Binary
---	0	000
--X	1	001
-W-	2	010
-WX	3	011
r--	4	100
r-X	5	101
rw-	6	110
rwX	7	111

```

$ echo `test` > test
$ ls -l test
$ chmod 777 test
$ ls -l test
$ chmod 755 test
$ ls -l test
    
```

- **Examples of chmod's mnemonic syntax (+/-/= add/remove/give)**

Spec	Meaning
u+w	Adds write permission for the owner of the file
ug=rw,o=r	gives r/w permission to owner and group, and read permission to others (,사이의 공백은 허용되지 않음)
a-x	Removes execute permission for all categories (owner/group/other)
g=u	Makes the group permissions be the same as the owner permissions

- User Type : u – owner, g – group, o – others, a – all (u,g and o)
- chmod --reference=filea fileb : make fileb's mode the same as filea's
- chmod -R g+w mydir : update permissions within a directory recursively 즉, Permission을 변경하고자 하는 대상이 디렉토리인 경우 그 하위 경로에 존재하는 모든 파일과 디렉토리의 Permission을 변경한다.

```

$ chmod a-x test
$ ls -l test
    
```

2. Process Handling

- **Job Control**

Job : A Process is usually referred to as a "job" when used in conjunction with job control .

Jobs are always in one of three states. running in the foreground, running in the background, or suspended.

- **foreground job** : There can be only one job in the foreground at any time. The foreground job has control of the shell with which you interact - it receives input from the keyboard and sends output to the screen.
- **background job** : Jobs in the background do not receive input from the terminal, generally running along quietly without the need for interaction. (started by appending a '&' character to the command line). Suspended job(temporarily stopped job) is also background job.

➤ **jobs** : the command to see list jobs

```
$ sleep 100 &
$ sleep 100 &
$ sleep 100 &
$ jobs
[1]  Running          sleep 100 &
[2]- Running          sleep 100 &
[3]+ Running          sleep 100 &
```

- Job number : refer to background processes that are currently running under shell

- **fg** : brings a background job into the foreground or resume suspended job in foreground
- no argument : pick the one that you put into the background most recently (= fg %+)
 - fg %1 : pick the one has job number '1'

```
$ jobs
[1]  Running          sleep 100 &
[2]- Running          sleep 100 &
[3]+ Running          sleep 100 &
$ fg
sleep 100
```

➤ **Ctrl-Z** : sends TSTP (terminal stop) signal to suspend a job

```
$ sleep 100
 (press Ctrl-Z)
[1]+  Stopped          sleep 100
$ jobs
[1]+  Stopped          sleep 100
```

**4190.102A Computer Programming
(2007 Winter)**

- **bg** : resume suspended job in background

```
$ sleep 100
(prompt Ctrl-Z)
[1]+  Stopped                  sleep 100
$ bg
[1]+  sleep 100 &
$ jobs
[1]+  Running                  sleep 100
```

- **ps**

ps : display a list of currently running processes.

```
$ ps
  PID TTY          TIME CMD
 9249 pts/2    00:00:00 bash
 9593 pts/2    00:00:00 ps
```

Process ID(PID) : a unique number given to every running process.

ps -aux : display all processes of system.

```
$ ps -aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START
TIME COMMAND
root           1  0.0  0.0   1944    128 ?        Ss   Nov16   0:45
init [2]
...
```

- **Signals**

A message that one process sends to another.

- Some abnormal event takes place
- It wants the other process to do something

```
~$ kill -l
1) SIGHUP          2) SIGINT          3) SIGQUIT        4) SIGILL
5) SIGTRAP        6) SIGABRT        7) SIGBUS         8) SIGFPE
9) SIGKILL        10) SIGUSR1       11) SIGSEGV       12) SIGUSR2
13) SIGPIPE       14) SIGALRM       15) SIGTERM       16) SIGSTKFLT
17) SIGCHLD       18) SIGCONT       19) SIGSTOP       20) SIGTSTP
21) SIGTTIN       22) SIGTTOU       23) SIGURG        24) SIGXCPU
25) SIGXFSZ       26) SIGVTALRM     27) SIGPROF       28) SIGWINCH
29) SIGIO         30) SIGPWR        31) SIGSYS        34) SIGRTMIN
...
```

- **Control-Key Signals**

Type	Signal
CTRL-C	INT (interrupt)
CTRL-Z	TSTP (terminal stop; to suspend the current job)
CTRL-\	QUIT ("stronger" version of INT)

- **Kill**
Sends the TERM (terminate) signal.

```
$ sleep 100 &  
$ sleep 100 &  
$ sleep 100 &  
$ jobs  
[1]  Running          sleep 100 &  
[2]- Running          sleep 100 &  
[3]+ Running          sleep 100 &  
$ kill %1  
[1]  종료됨           sleep 100  
$ jobs  
[2]- Running          sleep 100 &  
[3]+ Running          sleep 100 &
```

kill [-signal_name] [process_ID | %job#] : sends another signals.

```
$ sleep 100 &  
[1] 9609  
$ kill -9 9609  
[1]+ 죽었음           sleep 100
```

3. Supplementary commands

- **Hard and soft links**

- **In** : 파일을 실제 경로가 아닌 사용하기 편리한 다른 경로로 접근할 수 있도록 링크 생성.
- **Symbolic link (Soft link)**
 - 일반적인 링크 (Windows의 바로가기와 비슷)
 - 불필요한 파일의 복사를 하지 않아도 됨
 - 디렉터리까지 링크됨 (Hard link는 불가능)
 - 여러 디렉터리에서 동일한 라이브러리를 요구하거나 하나의 파일을 여러 사용자가 공통으로 사용할 경우 사용

4190.102A Computer Programming
(2007 Winter)

```
$ echo i am a student > student
$ ln -s student slink
$ ls -al
...
lrwxrwxrwx 1 comp-ta comp-ta    7 2008-01-02 07:20 slink ->
student
-rw-r--r-- 1 comp-ta comp-ta   15 2008-01-02 07:19 student
$ cat slink
$
$ ln -s /usr/bin bin
$ cd bin
$ ls -al
```

➤ Hard link

- 하드링크 된 파일은 원본과 동일하게 변경
- 원본이 삭제되어도 원본과 동일한 내용을 유지하므로 자원을 공유하되 데이터를 안전하게 관리하고자 할 때 사용

```
$ ln student hlink
$ ls -al
...
-rw-r--r-- 2 comp-ta comp-ta   15 2008-01-02 07:19 hlink
lrwxrwxrwx 1 comp-ta comp-ta    7 2008-01-02 07:20 slink ->
student
-rw-r--r-- 2 comp-ta comp-ta   15 2008-01-02 07:19 student
$ cat hlink
$ echo 'hahaha' >> student
$ cat hlink
$
$ rm student
$ ls -al
...
-rw-r--r-- 1 comp-ta comp-ta   22 2008-01-02 07:43 hlink
lrwxrwxrwx 1 comp-ta comp-ta    7 2008-01-02 07:20 slink ->
student
$ cat hlink
$ cat slink
```

● Finding files

- **find** : 주어진 조건에 따라 디렉터리의 트리를 검색하여 해당되는 파일을 찾음
 - 여러 가지 옵션(name, user, size...) 가능

```
$ find /home -name "test*" -print
$ find /home -name "test*" -print 2>/dev/null
$ find /home -user "comp-ta1" -print 2>/dev/null
$ find /home -size "-10k" -print 2>/dev/null
```

- **locate** : 자체 데이터베이스를 통해 파일을 찾아줌
 - 빠른 파일 찾기 가능

```
$ find / -name "httpd.conf" -print 2>/dev/null  
^C  
$ locate httpd.conf
```

- **File Compression and backup**

- **tar** : 여러 파일, 디렉터리를 하나의 파일로 묶는다 (tape archiver)
 - 압축을 수행하지 않고 묶는 역할만 함

```
(tar_test1 파일, tar_test2 파일, tar_test 디렉토리 생성)  
$ tar -cvf tarfile.tar tar_test1 tar_test2 (묶기)  
$ tar -cvf tarfile.tar ./tar_test (묶기)  
$ tar -tvf tarfile.tar (내용보기)  
$ mkdir test  
$ cp tarfile.tar ./test  
$ cd test  
$ tar -xvf tarfile.tar (풀기)  
$ ls -al
```

- **gzip** : Linux에서 가장 보편적으로 사용되는 압축방식
 - 주로 tar 명령과 함께 쓰임 (archive+compression)
 - tar의 옵션부분에 z 추가

```
$ tar -cvzf compressed_file.tar.gz ./test  
$ ls -al  
$ tar -xvzf compressed_file.tar.gz  
$ ls -al
```

4. Vi Editor

- **What is vi?**

- Unix와 Linux에서 가장 널리 쓰이는 텍스트 에디터
- small & powerful

- **Vi 실행**

- 새 문서 열기

```
$ vi 새 파일 이름
```

- 편집 문서 열기

```
$ vi 파일명 : 문서 열기  
$ vi -R 파일명 : 읽기 전용으로 문서 열기
```

➤ 문서 저장 / 끝내기

:w(!) : 문서 저장 (강제)
 :q(!) : vi 종료 (강제)
 :wq(!), zz(!) : 저장 후 종료 (강제)

● Modes

1. Command mode
 - Default mode(vi 파일명)
 - Move cursor, cut, paste, multi-window, searching, etc.
 - Case sensitive
2. Insert mode
 - Activated when you type "i"
 - You can edit the file as if you were using Windows notepad
 - If you want to back to normal mode, type "ESC"
3. Last line mode
 - Activated when you type ":"
 - Save, open, ... you can do everything except editing
 - 한글, MS-WORD의 '메뉴'에 해당
 - 다양한 옵션 가능 (:set nu)

● 주요 명령

참고 : 많은 명령어들이 (숫자)(명령어)형식으로 입력하면 해당 명령을 지정한 횟수만큼 반복하여 실행하는 기능이 있다.

커서 이동 명령	
h, j, k, l	좌, 하, 상, 우 로 커서 이동.
4l	오른쪽으로 4칸 이동
3k	위로 3행 이동
w	오른쪽으로 한 단어 이동
b	왼쪽으로 한 단어 이동
3w	오른쪽으로 3단어 이동
6b	왼쪽으로 6단어 이동
66G	66라인으로 이동
G	파일 맨 끝으로 이동

4190.102A Computer Programming
(2007 Winter)

문자, 행 삽입 명령 (Insert mode로 전환)	
a	현재 프롬프트 한 칸 오른쪽 위치에서 입력모드 전환
o	현재 행 다음 줄 삽입 후 입력모드 전환
O	현재 행 위에 줄 삽입 후 입력모드 전환
R	수정 모드(REPLACE)로 입력모드 전환

편집 명령	
~	대, 소문자 변경
u	Undo
Ctrl+r	Redo
U	행 전체 undo
:u	행 전체 redo
x	커서가 있는 문자 삭제
5x	커서가 있는 위치부터 5개의 문자를 삭제
dw	현재 커서가 있는 한 단어 삭제
db	커서의 위치에서 꺼꾸로 한 단어 삭제
dd	커서가 있는 라인 삭제
5dd	커서가 있는 라인부터 5개의 라인 삭제
:5,10d	5-10번째 행 삭제
yy	한 행 복사
6y	커서 아래 6행 복사
P	현재 행 위에 붙여넣기
p	현재 행 아래에 붙여넣기

행 번호 설정	
:set nu	행 번호 표시
:set nonu	행 번호 숨기기

검색	
/검색어	검색어를 찾아서 표시
n	이전 검색 위치로 이동
N	다음 검색 위치로 이동
:s/검색어/수정어	현재 라인의 첫 번째 검색어를 수정어로 바꿔줌
:%s/검색어/수정어/g	파일 전체에서 검색어를 수정어로 바꿔줌

<code>:%s/검색어/수정어/gc</code>	파일 전체에서 검색어를 확인 후 수정어로 바꿔줌
-----------------------------	----------------------------

Visual Mode	
<code>v</code>	Start visual mode (char-by-char)
<code>V</code>	Start visual mode (line-by-line)
<code>Esc</code>	End visual mode
<code>y</code>	복사
<code>p</code>	붙여넣기
<code>d</code>	삭제
<code>></code>	shift right
<code><</code>	shift left

- **실습**

```
$ ps aux > test.txt  
$ vi test.txt
```

위의 명령어들을 실습

- **참고자료**

- http://kldp.org/KoreanDoc/html/Vim_Guide-KLDP/Vim_Guide-KLDP.html
- <http://docs.freebsd.org/44doc/usd/12.vi/paper.html>