리눅스 기본 명령어 및 Vi 명령어 정리

2015/1, 컴퓨터의 개념 및 실습

리눅스 기본 명령어

리눅스 기본 명령어

- **새로운 파일을 만드는 방법**

(예)
vi newfile : vi 편집기 상태로 들어감
touch newfile : 빈 파일만 생성됨
cat > newfile : vi 편집기 상태로 들어감, 문서 작성후 Ctrl+D로 빠져나옴

- **파일 내용만 보기**

(예)
cat filename : 파일의 내용을 모두 보여줌
head -n filename : n줄 만큼 위세서부터 보여줌
tail -n filename : n줄 만큼 아래에서부터 보여줌

리눅스 기본 명령어

- **mv – 파일이름(rename) / 위치(move)변경**

  (예)
  # mv index.htm index.html : index.htm 화일을 index.html 로 이름 변경
  # mv file ../main/new_file : 파일의 위치변경

- **mkdir - 디렉토리 생성**

  (예)
  # mkdir download : download 디렉토리 생성

- **rm – 파일 삭제**

  (예)
  # rm test.html : test.html 화일 삭제
  # rm -r <디렉토리> : 디렉토리 전체를 삭제
  # rm -i a.* : a로 시작하는 모든 파일을 일일이 삭제할 것인지 확인하면서 삭제

- **rmdir – 디렉토리 삭제**

  (예)
  # rmdir cgi-bin : cgi-bin 디렉토리 삭제

리눅스 기본 명령어

- **ls – 파일 리스트 보기**

  F : 파일 유형을 나타내는 기호를 파일명 끝에 표시
  (디렉토리는 '/', 실행파일은 '*', 심볼릭 링크는 '@'가 나타남).
  l : 파일에 관한 상세 정보를 나타냅니다.
  a : dot 파일(.access 등)을 포함한 모든 파일 표시.
  t : 파일이 생성된 시간별로 표시
  C : 도스의 dir/w명령과 같 이 한줄에 여러개의 정보를 표시
  R : 도스의 dir/s 명령과 같이 서브디렉토리 내용까지.

  (예)
  # ls -al
  # ls -aC
  # ls -R

- **cd – 디렉토리를 변경**

  (예)
  # cd cgi-bin : 하부 디렉토리인 cgi-bin으로 들어감.
  # cd .. : 상위디렉토리로 이동
  # cd 또는 cd ~ : 어느곳에서든지 자기 홈디렉토리로 바로 이동
  # cd /webker : 현재 작업중인 디렉토리의 하위나 상위 디렉토리가 아닌 다른 디렉토리(webker)로 이동하려면 /로 시작해서 경로이름을 입력하면 된다.

리눅스 기본 명령어

- **pwd – 현재의 디렉토리 경로를 보여주기**

- **cat -** 파일의 내용을 화면에 출력하거나 파일을 만드는 명령

  (예)
  # cat filename

- **find – 각종 파일/디렉토리 검색하기**

  (예)
  # find –name '*.pl' : 현재 디렉토리에서 pl확장자를 가진 모든 파일 찾기
  # find / -name '*.pl' : 루트에서부터 pl확장자를 가진 모든 파일 찾기
  # find / -name 'ab*' : 루트에서부터 파일명이 ab로 시작하는 모든 파일 찾기
  # find / -name 'et*' –type d : 루트에서부터 디렉토리 이름이 et로 시작하는 모든 디렉토리 찾기

리눅스 기본 명령어

- **grep -** 파일 내에서 지정한 패턴이나 문자열을 찾기

사용법 :
Grep [-옵션] 패턴 파일명

옵션:
-c : 패턴이 일치하는 행의 수를 출력
-i : 비교시 대소문자를 구별 안함
-v : 지정한 패턴과 일치하지 않는 행만 출력
-n : 행의 번호를 함께 출력
-l : 패턴이 포함된 파일의 이름을 출력
-w : 패턴이 전체 단어와 일치하는 행만 출력

(예)
grep –n ‘hello’ example.txt : example.c 파일내에서 ‘int’ 패턴이 들어간 문자열과 행번호 출력
grep –l ‘hello’ * : 현재 디렉토리의 모든 파일에서 ‘hello’라는 패턴이 들어간 파일의 이름을 출력
grep –v ‘hello’ * : 현재 디렉토리의 모든 파일에서 ‘hello’라는 패턴이 들어가지 않은 행을 출력
grep –r ‘hello’ * : 현재 디렉토리의 및 서브디렉토리의 모든 파일에서 ‘hello’라는 패턴이 들어간 문자열 출력

리눅스 기본 명령어

- **grep의 패턴 정규 표현식 (Regular Expression)**

(예)
grep '^a' 파일명 : ^는 파일의 시작을 나타냄. 파일에서 a로 시작하는 행을 찾는다.
grep 'apple$' 파일명 : $는 파일의 끝을 나타냄. 파일에서 e로 끝나는 행을 찾는다.
grep 'app*' 파일명 : 파일에서 app로 시작하는 모든 단어를 찾는다.
grep 'a.....e' 파일명 : 파일에서 a로 시작하고 e로 끝나는 7자리 단어를 찾는다.
grep [a-d] 파일명 : 파일에서 a,b,c,d 로 시작하는 단어를 모두 찾는다.
grep [aA]pple 파일명 : 파일에서 apple 또는 Apple로 시작하는 단어를 모두 찾는다.
grep 'apple' d* : d로 시작하는 모든 파일에서 apple 를 포함하는 모든 행을 찾는다.
grep 'apple' 파일명1 파일명2 : 지정된 두개의 파일에서 apple 를 포함하는 모든 행을 찾는다.
grep '^[ab]' 파일명 : 파일에서 a나 b로 시작되는 모든 행을 찾는다.

리눅스 기본 명령어

- **man – 명령어의 모든 정보**

  (예)

  ```
  # man cd
  ```

Vi 기본 명령어

Vi 기본 명령어

- **저장 및 종료**

명령어	설명
:w	저장
:w file.txt	file.txt 파일로 저장
:w » file.txt	file.tx파일에 덧붙여서 저장
:q	vi 종료
:q!	vi 강제 종료
ZZ	저장 후 종료
:wq!	강제 저장 후 종료
:e file.txt	file.txt파일을 불러옴
:e	현재 파일을 불러옴
:e#	바로 이전에 열었던 파일을 불러 옴

Vi 기본 명령어

- **입력모드 전환**

a	커서 위치 다음칸부터 입력	A	커서 행의 맨 마지막부터 입력
i	커서의 위치에 입력	I	커서 행의 맨 앞에서 부터 입력
o	커서의 다음행에 입력	O	커서의 이전 행에 입력
s	커서 위치의 한글자를 지우고 입력	cc	커서위치의 한 행을 지우고 입력

Vi 기본 명령어

- **이동**

h	왼쪽으로 이동	**l**	오른쪽으로 이동
j	아래행으로 이동	**k**	위 행으로 이동
w 또는 W	다음 단어의 첫 글자로 이동	**b 또는 B**	이전 단어의 첫 글자로 이동
e 또는 E	단어의 마지막 글자로 이동	**<CR>**	다음행 첫 첫 글자로 이동
^	그행의 첫 글자로 이동	**$**	그 행의 마지막 글자로 이동
+	다음 행의 첫 글자로 이동	**-**	위 행의 첫 글자로 이동
(	이전 문장의 첫 글자로 이동	**)**	다음 문장의 첫 글자로 이동
{	이전 문단으로 이동	**}**	다음 문단으로 이동
H	커서를 화면 맨 위로 이동	**z<CR>**	현재 행을 화면의 맨우로 이동
M	커서를 화면 중안으로 이동	**z.**	현재 행을 화면의 중앙으로 이동
L	커서를 화면 최하단으로 이동	**z-**	현재 행의 화면의 최하단으로 이동
[n]H	커서를 위에서 n행으로 이동	**[n]L**	커서를 아래에서 n행으로 이동
ctrl+u	반 화면 위로 스크롤	**ctrl+d**	반 화면 아래로 스크롤
ctrl+b	한 화면 위로 스크롤	**ctrl+f**	한 화면 아래 스크롤
gg 또는 1G	문서의 맨 처음으로 이동	**G**	문서의 맨 마지막 행으로 이동
[n]G 또는 :[n]	n행으로 이동		

Vi 기본 명령어

- **삭제**

x 또는 dl	**커서 위치의 글자 삭제**
X 또는 dh	커서 바로 앞의 글자 삭제
dw	현재 위치부터 스페이스 까지 삭제
diw	**현재 위치에 있는 단어 삭제**
dd	**커서가 있는 행을 삭제**
[n]dd	현재 커서 부터 아래 **n**번째 줄까지 삭제
dj	현재 커서와 아래 줄 삭제
[n]dj	현재 커서 부터 아래 **n+1**번째 줄까지 삭제
dk	현재 커서와 윗로 **n+1**번째 줄까지 삭제
[n]dk	현재 커서와 윗 줄 삭제
D 또는 d$	현재 커서가 있는 위치부터 행 끝까지 삭제
d0 또는 d^	현재 커서가 있는 위치부터 행 시작 까지 삭제

Vi 기본 명령어

- **복사 & 붙여넣기**

yy 또는 **Y**	커서가 있는 한 행 복사
p	현재 커서에 붙여 넣기, 행 복사 일 경우 아래 줄에 붙여넣음.
P	현재 커서위치의 앞행에 붙여 넣기, 행 복사일 경우에는 윗 줄에 붙여 넣음
[n]yy 또는 **[n]Y**	커서가 위치한 이후로 **n**행 복사
[n]p	**n**번 만큼 붙여넣기 반복

- **블록 지정**

v	블록 지정
V	줄단위 블록 지정
ctrl+v(윈도우에서는 ctrl+q)	비쥬얼 블록 지정
블록 지정 중 명령	
y	블록 복사 하기
r	치환
d	지정 블록 지우기
U	대문자로 바꾸기
u	소문자로 바꾸기
~	대소문자 전환
J	행 합침
:	선택 영역에 대하여 ex명령
<	행 앞에 탭 제거
>	행 앞에 탭 삽입

Vi 기본 명령어

- **다중 창 관련 명령**

명령모드	ex모드	결과
창생성		
CTRL-W s	:[N]sp[plit]	현재 파일을 두 개의 수평 창으로 나눔
CTRL-W v	:[N]vs[plit]	현재 파일을 두 개의 수직 창으로 나눔
CTRL-W n	:new	새로운 수평 창 생성
CTRL-W ^ 또는 CTRL-W CTRL-^		수평 창으로 나누고 이전 파일의 오픈
CTRL-W f		창을 수평으로 나누고 커서 위치의 파일 오픈
CTRL-W i		커서 위치의 단어가 정의된 파일을 오픈
창삭제		
CTRL-W q	:q[uit]!	현재 커서의 창을 종료
CTRL-W c	:close	현재 커서의 창 닫기
CTRL-W o	:on[ly]	현재 커서의 창만 남기고 모든 창 삭제
창이동		
CTRL-W h		왼쪽 창으로 커서 이동
CTRL-W j		아래쪽 창으로 커서 이동
CTRL-W k		위쪽 창으로 커서 이동
CTRL-W l		오른쪽 창으로 커서 이동
CTRL-W w		창을 순차적으로 이동
CTRL-W p		가장 최근에 이동한 방향으로 이동
CTRL-W t		최상위 창으로 이동
CTRL-W b		최하위 창으로 이동
창이동		
CTRL-W r		순환으로 창의 위치를 순환
CTRL-W x		이전 창과 위치를 바꿈
CTRL-W H		현재창을 왼쪽 큰화면으로 이동
CTRL-W J		현재창을 아래쪽 큰화면으로 이동
CTRL-W K		현재창을 위쪽 큰화면으로 이동
CTRL-W L		현재창을 오른쪽 큰화면으로 이동
창 크기 조정		
CTRL-W =		창의 크기를 모두 균등하게 함
CTRL-W _		수평 분할에서 창의 크기를 최대화
CTRL-W |		수직 분할에서 창의 크기를 최대화
CTRL-W [N]+	:res[ize] +N	창의 크기를 N행 만큼 증가
CTRL-W [N]-	:res[ize] -N	창의 크기를 N행 만큼 감소
CTRL-W [N]>		창의 크기를 오른쪽으로 N칸 만큼 증가
CTRL-W [N]<		창의 크기를 오른쪽으로 N칸 만큼 감소

Vi 기본 명령어

- **문자열 찾기**

명령어	동작
/name	name 문자열 찾기
n	다음 name으로 이동
N	n과 같으며 역방향으로 이동

- **문자열 대체**

명령어	동작
:s/str/rep	현재 행의 str을 rep로 대체
:1,.s/str/rep/	1부터 현재 행의 str을 rep로 대체
:%s/str/rep/g	파일 전체 str을 rep로 전부 대체
:.,$/aaa/bbb	커서의 위치로부터 파일의 끝까지 있는 모든 aaa를 bbb로 대체

- **기타**

명령어	동작
:set nu	행 번호 보여주기
:set nonu	행 번호 보여주기 취소
.	바로 전에 실행한 명령어 재 실행
Ctrl + l	불필요한 화면 정리후 다시 표시

gcc 기본 명령어

gcc 기본 명령어

- **An example C Code, main.c**

```
#include<stdio.h>


int main(void)
{
   printf("\n The Geek Stuff\n");
   return 0;
}
```

1. gcc compiler can be used as below in most basic form

```
gcc main.c
```

gcc 기본 명령어

2. Specify the Output Executable Name

```
gcc main.c -o main
```

3. Enable all warnings set through –Wall option

```
$ gcc -Wall main.c -o main
```

4. Produce only the preprocessor output with –E option

```
$ gcc -E main.c > main.i
```

gcc 기본 명령어

5. Produce only the assembly code using –S option

```
gcc -S main.c > main.s
```

6. Produce only the compiled code using the –C option

```
gcc -C main.c
```

7. Produce all the intermediate files using –save-temps function

```
$ gcc -save-temps main.c

$ ls
a.out  main.c  main.i  main.o  main.s
```

gcc 기본 명령어

8. Link with shared libraries using –l option

```
gcc  -Wall main.c -o main -lCPPfile
```

9. Print all the executed commands using –V option

```
$ gcc -Wall -v main.c -o main
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/i686-linux-gnu/4.6/lto-wrapper
Target: i686-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.6.3-1ub
Thread model: posix
gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5)
```