LINUX 개발환경(1)

- LINUX Shell 개요
- LINUX 텍스트 편집기

Computer & Software Engineering
Lee Jong Yeol
soyeum@kumoh.ac.kr

LINUX Shell 개요

- Shell ?
  - Kernel과 사용자 간의 인터페이스 역할 수행
  - 사용자로부터 명령을 받아 그것을 해석하고 프로그램을 실행
- Shell 기능
  - 명령어 해석기 기능(Interpreter/Translator)
    - 사용자 명령을 해석하여 해당 프로그램에 전달
    - /etc/passwd에서 로그인하면 실행되는 'Login Shell'을 지정
  - 프로그래밍 기능
    - 반복적으로 수행하는 작업을 하나의 프로그램으로 작성 가능
    - 쉘 프로그램 = 쉘 스크립트
  - 사용자 환경 설정 기능
    - 초기화 파일(.<쉘이름>rc, .profile, .config 등)로 사용자 환경 설정

LINUX Shell 개요

❑ Shell 종류(1)

- 본 쉘(Bourne shell): sh
  - 유닉스 V7에 등장한 최초의 쉘
  - 개발자 스테판 본(Stephen Bourne)이름을 따서 본 쉘이라 함
  - 단순하고 처리속도가 빨라 시스템 관리 작업용 쉘 스크립트에 사용
  - history, alias, 작업 제어 등의 기능이 없어 다른 쉘들이 등장

- C 쉘(C shell): csh
  - 캘리포니아 버클리대학교 빌 조이(Bill Joy)가 개발
  - BSD 유닉스에 포함되어 발표
  - history, alias 등의 사용자 편의 기능 포함
  - 쉘 스크립트 구문 형식이 C 언어와 같아서 C 쉘이라 함

LINUX Shell 개요

❑ Shell 종류(2)

- 콘 쉘(Korn shell): ksh
  - AT&T 벨연구소의 데이비드 콘(David Korn)이 개발
  - 유닉스 SVR-4에 포함되어 발표
  - 본 쉘과 호환성을 유지하면서 history, alias 등의 기능 제공
  - 처리 속도가 상대적으로 빠름

- 배시 쉘(bash shell): bash
  - 본 쉘을 기반으로 브레인 폭스(Brain Fox)가 개발(1988)
  - bash = sh + csh + ksh
  - GPL 라이선스에 의해 자유롭게 사용 가능
  - 리눅스의 기본 쉘로 채택 됨

LINUX Shell 개요

❑ Shell 종류(3)

- 대시 쉘(dash shell): dash
  - 본 쉘을 기반으로 POSIX 표준을 준수하면서 상대적으로 작은 크기
  - 암키스트 쉘(ash, Almquist shell)의 NetBSD 버전으로 허버트 슈가 리눅스에 이식함
  - 우분투 6.10부터 본 쉘 대신에 대시 쉘을 사용 ($ ls -l /bin/sh 로 확인 가능)

- 기본 쉘(login shell) 확인: grep <사용자계정> /etc/passwd
  - 프롬프트 모양으로 확인
    $: sh, bash, ksh
    % : csh
  - /etc/passwd 파일에서 해당 사용자 정보의 가장 마지막 항목에서 확인

LINUX Shell 개요

❑ Shell 사용(1)

- 기본 쉘(login shell) 교체: chsh [옵션] [사용자명]

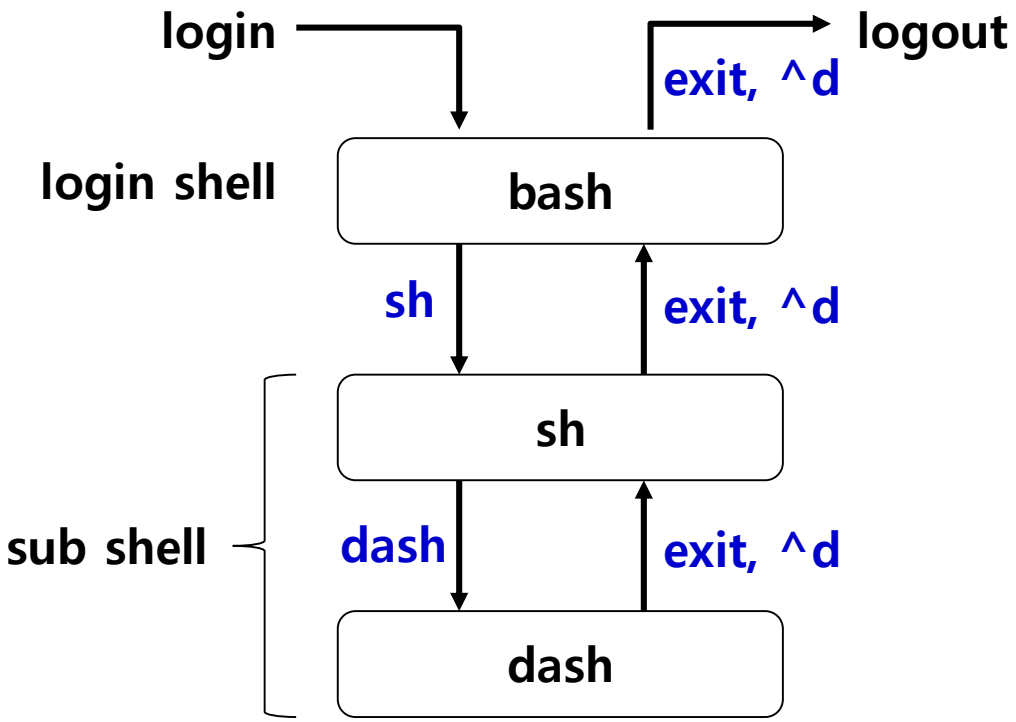
```
linuxer@linuxer-PC:~$ cat /etc/shells          (바꿀 수 있는 쉘 종류 확인)
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash          ← 바꾸려는 쉘은 항상 절대경로 사용
/usr/bin/rbash
/bin/dash
/usr/bin/dash

linuxer@linuxer-PC:~$ chsh -s sh linuxer
암호:
chsh: sh is an invalid shell
linuxer@linuxer-PC:~$ chsh -s /bin/sh linuxer
linuxer@linuxer-PC:~$ echo $SHELL          (자신의 쉘 확인)
```

LINUX Shell 개요

❑ Shell 사용(2)

▪ login shell vs. sub shell




```
linuxer@linuxer-PC:~$ sh
$ dash
$ exit
$ exit
linuxer@linuxer-PC:~$
```

LINUX Shell 개요

❑ Shell 사용(3)

▪ 쉘 명령 종류 확인: type 쉘명령

우선순위	명령 유형	설명
1	alias	특정 명령을 옵션과 함께 짧은 별칭으로 정의
2	쉘 예약어	쉘 스크립트 작성 시 사용되는 쉘 고유 키워드 ex> do, while, case, ...
3	함수	일련의 쉘 명령을 함수로 정의한 것
4	내장 명령	쉘 자체에 포함된 built-in 명령들 ex> cd, echo, pwd, printf, ...
5	일반 명령	파일 시스템(/usr/bin, /usr/sbin)에 실행 파일로 존재하는 명령들 ex> ls, grep, man, cal, ...

```
linuxer@linuxer-PC:~$ type cd
cd is a shell builtin
linuxer@linuxer-PC:~$ type case
case is a shell keyword
```

```
linuxer@linuxer-PC:~$ type ls
ls 은/는 `ls --color=auto' 의 별칭
linuxer@linuxer-PC:~$ type ll
ll 은/는 `ls -alF' 의 별칭
```

LINUX Shell 개요

❑ Shell 사용(4)

- **쉘 일반 명령 확인: file [옵션] 파일명**
  - **쉘 일반 명령은 실행 파일로 존재(/usr/bin, /usr/sbin)**
  - **실행 파일은 바이너리 파일이므로 cat 명령으로 파일 내용 확인 불가**

```
linuxer@linuxer-PC:~$ file /usr/bin/man
/usr/bin/man: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=1eb41e661c87bfd3d4d0d1d54cf2247a240c407e, for
GNU/Linux 3.2.0, stripped

linuxer@linuxer-PC:~$ file /usr/bin/grep
/usr/bin/grep: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=f4564437ed282494b69a191fcb07a56235cce8ff, for
GNU/Linux 3.2.0, stripped
```

LINUX Shell 개요

❑ Shell 사용(5)

▪ 쉘 메타 문자들

문자	의미
>	표준 출력을 파일에 기록하는 출력 리다이렉션 기호
>>	표준 출력을 파일의 끝에 추가하는 출력 리다이렉션 기호
<	파일로 부터 표준 입력을 읽어 들이는 입력 리다이렉션 기호
*	널 문자열을 포함한 모든 문자열로 치환
?	모든 단일 문자로 치환
[ ]	대활호 안의 어떠한 문자와도 일치하는 정규식 표현
|	표준 출력을 입력으로 보내는 파이프 기호
||	이전 명령이 실패하면 실행(조건부 실행)
;	명령어의 분리자 기호(연속 명령을 왼쪽부터 차례로 실행)
&	백그라운드 모드 실행
&&	이전 명령이 성공할 경우 실행(조건부 실행)
#	주석 처리
$	변수에 접근
-	cd 명령으로 디렉터리를 이전하기 직전의 작업 디렉터리 표시

LINUX Shell 개요

❑ Shell 사용(6)

▪ 쉘 환경 변수 확인

- echo $환경변수명
- printf $환경변수명
- printenv [환경변수명]
- set
- env
- export

```
$ echo $UID
1000
$ printenv PWD
/home/linuxer
$ set          (global 변수 확인)
$ env          (loval 변수 확인)
$ export
```

변수	의미
BASH	사용하고 있는 bash 경로
BASH_VERSION	사용하고 있는 bash 버전
COLUMNS	터미널의 행수(=80)
DISPLAY	현재 X-Window display 위치
HISTFILE	히스토리 파일명
HISTFILESIZE	히스토리 파일 사이즈
HISTSIZE	히스토리 개수
HOME	사용자의 홈 디렉터리
HOSTNAME	시스템의 호스트명
HOSTTYPE	시스템의 타입 값
LINES	터미널의 라인 수
LOGNAME	로그인 사용자명
OSTYPE	운영체제의 타입
PS1	주 프롬프트 문자열 설정값
PATH	명령을 찾을 검색 경로
UID	현재 사용자의 UID 값

LINUX Shell 개요

❑ Shell 사용(7)

▪ 쉘 환경 변수값 설정 및 해제

- 쉘 변수 설정: 변수명=문자열
- export [환경변수명]=[변수값]
- unset 변수명

```
linuxer@linuxer-PC:~$ PGMDIR=/home/linuxer/src
linuxer@linuxer-PC:~$ echo $PGMDIR                (설정 확인)

linuxer@linuxer-PC:~$ export                      (설정 출력)
linuxer@linuxer-PC:~$ export myvar                (변수 설정)
linuxer@linuxer-PC:~$ export myvar=100            (값 설정)
linuxer@linuxer-PC:~$ echo $myvar                 (설정 확인)

linuxer@linuxer-PC:~$ export myvar=200            (값 설정)
linuxer@linuxer-PC:~$ echo $myvar                 (설정 확인)

linuxer@linuxer-PC:~$ unset myvar                 (값 설정)
linuxer@linuxer-PC:~$ echo $myvar                 (설정 확인)
```

LINUX Shell 개요

❑ Shell 사용(8)

- 별명(alias) 설정: alias [이름='명령']

```
linuxer@linuxer-PC:~$ alias                          (설정된 alias 표시)

linuxer@linuxer-PC:~$ alias list='ls -alF'           (alias 설정)
linuxer@linuxer-PC:~$ list                           (alias 사용)

linuxer@linuxer-PC:~$ alias rm='rm -i'               (alias 설정)
```

- 별명(alias) 해제: unalias 별명

```
linuxer@linuxer-PC:~$ unalias  list
linuxer@linuxer-PC:~$ unalias  rm

linuxer@linuxer-PC:~$ unalias  list  rm
```

LINUX Shell 개요

❑ Shell 사용(9)

▪ 별명(alias)에 인자 전달

- 인자는 쉘 함수를 작성해서 전달
- 인자 전달 순서를 인자명으로 지정: $1 $2 $3 …
- $0 는 실행된 스크립트 이름

```
linuxer@linuxer-PC:~$ function mycd {
> cd $1; pwd;
> }
linuxer@linuxer-PC:~$ mycd /etc
/etc
linuxer@linuxer-PC:/etc$
```

LINUX Shell 개요

❑ Shell 사용(10)

▪ 히스토리(history) 확인: history

- 쉘에 입력된 명령들은 로그아웃할 때 ~/.bash_history 파일에 기록됨
- 키보드 상(↑), 하(↓) 키를 이용해서 이전에 입력된 명령을 호출 가능
- 아래와 같은 방법으로 이전의 명령을 재실행 가능

입력	기능
!!	바로 직전에 실행한 명령을 재실행
!번호	히스토리에서 해당 번호의 명령을 재실행
!문자열	히스토리에서 해당 문자열로 시작하는 마지막 명령을 재실행

```
linuxer@linuxer-PC:~$ history
linuxer@linuxer-PC:~$ ls -ld .?*                    (숨김 파일만 표시)
linuxer@linuxer-PC:~$ !!
linuxer@linuxer-PC:~$ !ls
linuxer@linuxer-PC:~$ tail -n 10 .bash_history
```

LINUX Shell 개요

❑ Shell 사용(11)

▪ 프롬프트(prompt) 변경

- 쉘 환경변수 PS1에 새로운 형태의 문자열을 지정
- 프롬프트에서 사용할 수 있는 이스케이프 문자 활용 (색상 지정 가능)

```
linuxer@linuxer-PC:~$ PS1='[$PWD]'
[/home/linuxer] cd ..
[/home] PS1='[₩u ₩T] (₩!) $ '
[linuxer 14:38:01] (164) $
```

문자	기능
\a	ASCII 벨 문자
\d	'요일 월 일' 형식
\h	호스트명
\H	전체 호스트명
\s	쉘 이름
\t	HH:MM:SS(24시간)
\T	HH:MM:SS(12시간)

문자	기능
\@	현재시간(오전/오후)
\u	사용자 이름
\v	배시 쉘 버전
\w	현재 작업 디렉터리
\W	최종 작업 디렉터리
\!	히스토리 번호
\n	줄 바꾸기

LINUX Shell 개요

❑ Shell 환경 설정 파일(1)

- 환경 설정 파일
  - 사용자가 로그인할 때마다 자동으로 실행되는 명령을 저장
  - 쉘 마다 다른 이름의 파일을 사용
- 시스템 환경 설정 파일
  - 시스템을 사용하는 전체 사용자의 공통 환경

파일	기능
/etc/profile	• 본 쉘이나 호환되는 모든 쉘에 공통으로 적용(.profile) • bash은 /etc/bash.bashrc 파일 실행 • /etc/profile.d/*.sh 파일을 실행
/etc/bash.bashrc	• 시스템 공통으로 적용되는 .bashrc 파일 • 기본 프롬프트를 설정 • sudo 명령과 관련된 힌트를 설정
/etc/profile.d/*.sh	• 언어나 명령별로 각각 필요한 환경을 설정 • 필요시 설정 파일을 추가 가능

LINUX Shell 개요

❑ Shell 환경 설정 파일(2)

▪ 사용자 환경 설정 파일

- 각 사용자의 홈 디렉터리에 숨김 파일로 생성
- 사용자가 내용을 직접 수정하고 관리 가능

파일	기능
~/.profile	• 경로 추가 등 사용자가 정의하는 환경을 설정 • .bashrc 파일이 있으면 실행
~/.bashrc	• 히스토리의 크기를 설정 • 기본 alias나 함수 등을 설정
~/.bash_aliases	• 사용자가 정의한 alias를 별도 파일로 저장
~/.bash_logout	• 로그아웃 시 실행할 필요가 있는 함수 등을 설정

LINUX Shell 개요

❑ Shell 환경 설정 파일(3)

▪ 다른 쉘의 환경 설정 파일

쉘	시스템 초기화 파일	사용자 초기화 파일	실행 시기		
			로그인	서브 쉘	로그아웃
본 쉘 (sh)	**/etc/profile**	**$HOME/.profile**	○		
콘 쉘 (ksh)	**/etc/profile**	**$HOME/.profile**	○		
		$HOME/.kshrc	○	○	
C 쉘 (csh)	**/etc/.login**	**$HOME/.login**	○		
		$HOME/.cshrc	○	○	
		$HOME/.logout			○

LINUX 텍스트 편집기

❑ 리눅스 편집기 종류

- 행 단위 편집기: ed, ex, sed
- 화면 단위 편집기: vi, emacs
- GUI 기반 편집기: gedit

❑ ed 편집기 사용

- 첫 문자가 명령: i, d, w, q, …

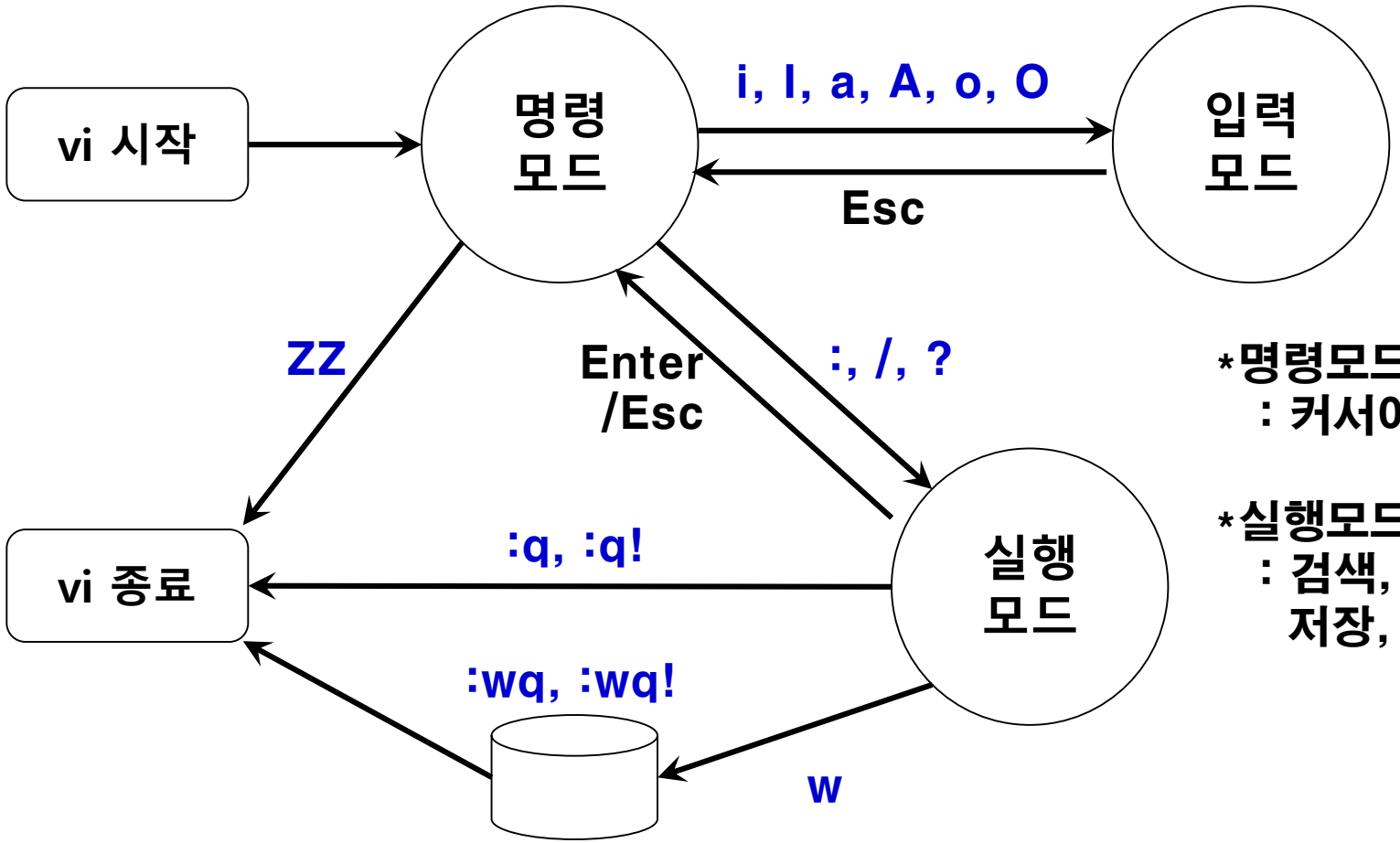
```
linuxer@linuxer-PC:~$ ed
i                       (행 단위 입력 시작)
first line
second line
.                       (행 단위 입력 종료)
w ed.txt                (입력 내용을 파일로 저장)
q                       (ed 프로세스 종료)
[linuxer 14:38:01] (164) $
```

LINUX 텍스트 편집기

- vi 편집기
  - 화면 단위로 편집하는 Visual Editor
  - ed 편집기를 개선하여 버클리 대학의 빌 조이(Sun 공동창업자)가 개발('76)
  - UNIX의 기본 텍스트 편집기로 제공
  - ed 편집기는 AT&T의 라이선스없이 코드 수정이 불가
  - vi를 오픈 소스화한 여러 vi 등장
  - 리눅스에서는 vi 기능을 향상시킨 vim (vi improved) 제공
  - 여러 IDE에 플러그인으로 vi 제공: VsVim, Xvim, Vrapper,...
  - 웹 브라우저 플러그인으로 vi 제공: Vimium(FireFox, Chrome), ...
  - 환경설정 파일 편집, 프로그래머의 코드 편집 용도로 주로 사용
  - 입력 모드, 명령 모드, 실행 모드로 구분하여 동작

LINUX 텍스트 편집기

❑ vi 편집기 동작 모드



*명령모드
: 커서이동, 내용수정

*실행모드
: 검색, 바꾸기,
저장, 종료

LINUX 텍스트 편집기

❑ 명령 모드(1) – 커서 이동 명령

명령 키	설명
h	커서를 왼쪽으로 한 칸 이동(←)
j	커서를 아래로 한 칸 이동(↓)
k	커서를 위로 한 칸 이동(↑)
l	커서를 오른쪽으로 한 칸 이동(→)
w	다음 단어의 처음으로 이동
^	줄의 첫 문자로 이동
$	줄의 맨 끝으로 이동
0	첫 번째 열로 이동
G	제일 끝 행으로 이동
gg	제일 첫 행으로 이동
nG	n번째 행으로 이동

명령 키	설명
H	화면의 첫 줄로 이동
M	화면 중간으로 이동
L	화면 끝 줄로 이동
Ctrl+b	이전 화면으로 이동
Ctrl+f	다음 화면으로 이동
Ctrl+d	반 정도 화면 이동
Ctrl+y	한 줄씩 내려감
Ctrl+e	한 줄씩 올라감
n%	퍼센트에 해당하는 줄로 이동
:숫자 +Enter	해당 숫자의 행으로 이동

LINUX 텍스트 편집기

❑ 명령 모드(2) – 내용 수정 명령

명령 키	설명
r	커서가 위치한 글자를 다른 글자로 수정
cw **#cw**	커서 위치부터 현재 단어의 끝까지 수정 #에는 수정할 단어의 수를 지정 (ex. 2cw는 두 단어만 수정)
s **#s**	커서 위치부터 Esc 키를 입력할 때까지 수정 #에는 수정할 글자의 수를 지정 (ex. 4s는 네 글자만 수정)
cc	커서가 위치한 행의 내용을 모두 수정
C	커서 위치부터 행의 끝까지 수정

LINUX 텍스트 편집기

❑ 명령 모드(3) – 내용 삭제 명령

명령 키	설명
x **#x**	커서 위치의 글자를 삭제 #에는 삭제할 글자 수를 지정
dw **#dw**	커서 위치의 단어를 삭제 #에는 삭제할 단어의 수를 지정
dd **#dd**	커서 위치의 행을 삭제 #에는 삭제할 행의 수를 지정
D	커서 위치부터 행의 끝까지 삭제

LINUX 텍스트 편집기

❑ 명령 모드(4) – 명령 취소 명령

명령 키	설명
u	명령을 취소
U	해당 행에서 한 모든 명령을 취소
:e!	마지막으로 저장한 내용 이후의 것을 버리고 새로 작업

LINUX 텍스트 편집기

❑ 명령 모드(5) – 복사, 잘라내기, 붙이기 명령

명령 키	설명
yy **#yy**	커서가 위치한 행을 복사 #에는 복사할 행의 수를 지정
p	커서가 위치한 행의 아래쪽에 붙임
P	커서가 위치한 행의 위쪽에 붙임
dd **#dd**	커서가 위치한 행을 잘라둠 (삭제와 같은 기능) #에는 잘라둘 행의 수를 지정

LINUX 텍스트 편집기

❑ 명령 모드(6) – 기타 명령

명령 키	설명
.	바로 직전에 했던 명령을 반복 수행
Shift+j	현재 행과 아랫행을 연결하여 한 행으로 묶음
Ctrl+l	현재 화면을 다시 출력
Ctrl+g	현재 커서 위치의 행 번호를 마지막 행에 출력

LINUX 텍스트 편집기

❑ 실행 모드(1) – 복사, 잘라내기, 붙이기 명령

명령 키	설명
:#y	#로 지정한 행을 버퍼로 복사
:<범위>y	범위로 지정한 행을 버퍼로 복사
:#d	#로 지정한 행을 삭제
:<범위>d	범위로 지정한 행을 삭제
:pu	현재 행 다음에 버퍼의 내용을 붙이기
:#pu	#로 지정한 행 다음에 버퍼의 내용을 붙이기

<범위>

명령 키	설명
1,$ 또는 1,%	1행부터 마지막 행까지 지정
1,.	1행부터 커서가 있는 행까지 지정
.,$	커서가 있는 행부터 마지막 행까지 지정
,-3	현재 행과 이전 세 행까지(총 네 행) 지정
10,20	10행부터 20행까지 지정

LINUX 텍스트 편집기

❑ 실행 모드(2) – 검색 명령

명령 키	설명
/문자열	문자열을 아래 방향으로 검색
?문자열	문자열을 위 방향으로 검색
n	원래 찾던 방향으로 다음 문자열 검색
N	반대 방향으로 다음 문자열 검색

LINUX 텍스트 편집기

❑ 실행 모드(3) – 바꾸기 명령

명령 키	설명
:s/문자열1/문자열2/	커서가 위치한 행에서 첫 번째로 나오는 문자열1을 문자열2로 교체
:%s/문자열1/문자열2/g	파일 전체에서 모든 문자열1을 문자열2로 교체
:<범위>s/문자열1/문자열2/	범위 내 모든 행에서 첫 번째로 나오는 문자열1을 문자열2로 교체
:<범위>s/문자열1/문자열2/g	범위내 모든 행에서 문자열1을 문자열2로 교체
:<범위>s/문자열1/문자열2/gc	범위내 모든 행에서 문자열1을 문자열2로 바꿀 때 수정할지 여부를 물으면서 교체

LINUX 텍스트 편집기

❑ 실행 모드(4) – 파일 명령

명령 키	설명
:r 파일명	지정한 파일을 읽어서 현재 커서 위치에 삽입
:w 파일명	지정한 파일로 저장 (동일 파일이 있다면 덮어쓰기 안됨)
:w! 파일명	지정한 파일로 강제 저장 (덮어쓰기 가능)
:e 파일명	지정한 파일로 전환 (기존 파일을 :w로 저장할 필요 있음)
:e! 파일명	지정한 파일로 강제 전환
:n	vi 시작 시 여러 파일을 지정했을 경우 다음 파일로 작업 이동
:n!	편집 작업 중인 현재 파일을 저장하고 다음 파일로 작업 이동

LINUX 텍스트 편집기

❑ 실행 모드(5) – 쉘 명령

명령 키	설명
:! <쉘명령>	vi 작업을 잠시 중단하고 쉘 명령을 실행 vi로 돌아오려면 Enter 키를 입력
:.! <쉘명령>	쉘 명령을 수행하고 그 결과를 현재 위치에 붙여넣음
:sh	vi를 잠시 빠져나가서 쉘 명령을 실행 vi로 돌아오려면 exit 명령 입력

LINUX 텍스트 편집기

❑ 실행 모드(6) – vi 설정 명령

명령 키	설명
:set	vi 편집기 현재 설정 상태 출력
:set all	vi 편집기의 모든 설정 값 출력
:set number	편집 중인 문서의 행 번호 출력 (= **:set nu**)
:set nonumber	편집 중인 문서의 행 번호 출력 중지 (= **:set nonu**)
:set tabstop	현재 설정된 탭문자의 공백수를 확인
:set tabstop=<값>	지정된 값으로 탭문자의 공백수를 설정
:set list	눈에 보이지 않는 특수문자를 표시
:set nolist	눈에 보이지 않는 특수문자 숨김
:set autoindent	자동으로 들여쓰기 설정 (= **:set ai**)
:set noautoindent	들여쓰기 설정 제거 (= :set noai)
:sh	vi를 잠시 빠져나가서 쉘 명령을 실행 vi로 돌아오려면 exit 명령 입력

LINUX 텍스트 편집기

❑ 실행 모드(7) – vi 설정 파일 생성

▪ .exrc 파일에 설정

- vi 실행할 때 자동으로 설정(set) 되도록 .exrc 파일 직접 생성
- 파일에는 set 명령과 옵션만 지정 가능
- 각 행을 주석 처리하려면 이중따옴표(")를 첫글자에 사용

```
linuxer@linuxer-PC:~$ vi .exrc
set nu
set ai
"set showmode
```

▪ EXINIT 환경 변수에 설정

```
linuxer@linuxer-PC:~$ EXINIT='set nu ai showmode'
linuxer@linuxer-PC:~$ export EXINIT
linuxer@linuxer-PC:~$
```