

ECE20023, Spring 2021



오픈소스 소프트웨어 실습



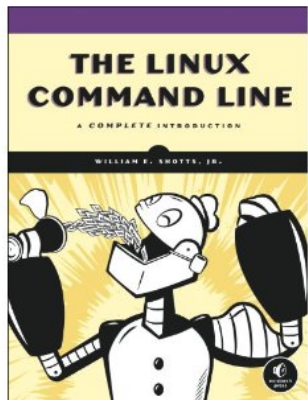


8

Vim Text editor

Shell Commands

Chap6 ~ chap10



- Chapter 5. Commands
 - type / which / help / man / apropos / info / whatis / alias
- Chapter 6. Redirection
 - cat / sort / uniq / grep / wc
 - head / tail / tee
- Chapter 7. Seeing
 - echo
- Chapter 8. Keyboard tricks
 - clear / history
- Chapter 9. Permissions
 - id / [chmod](#) / [umask](#) / su / sudo / chown / chgrp
- Chapter 10. Processes
 - ps / top / jobs / bg / fg / kill / killall / shutdown

Vim

- Vim = Vi improved
- vi 호환 text editor
- CUI용 Vim, GUI용 gVim
- Windows, linux, MacOS 등 다양한 버전 프로그램 지원

<https://www.vim.org/>

The screenshot shows the Vim.org homepage with a green header bar. The header contains links for 'SPONSOR', 'VOTE', 'the editor' (with a logo), 'BUY', 'HELP', and 'LEARN'. Below the header, there's a search bar with 'SEARCHED BY Google' and a 'Search' button. A sidebar on the left lists navigation links: Home, Advanced search, About Vim, Community, News, Sponsoring, Trivia, Documentation, Download (highlighted in green), Vim from GitHub, Vim from Mercurial, List of mirrors, Sources, Patches, Development, Runtime files, Script links, Translations, Old stuff, and Scripts. The main content area is titled 'Downloading Vim' and explains that Vim is available for many systems. It lists 'Most popular' download links for MS-Windows (self-installing executable, signed files), Unix (GitHub page, Mercurial, Aquimage), and Mac (MacVim project, Hemsikrew). A 'Details and options for:' section lists links for Unix, PC: MS-DOS and MS-Windows, Amiga, and OS/2.

SPONSOR VOTE
Vim development for features

the editor

BUY the Vim book HELP Uganda LEARN Vim

not logged in [login]

SEARCHED BY Google

Search

Home
[Advanced search](#)

About Vim
[Community](#)
[News](#)
[Sponsoring](#)
[Trivia](#)
[Documentation](#)
Download

Vim from GitHub
[Vim from Mercurial](#)
[List of mirrors](#)
[Sources](#)
[Patches](#)
[Development](#)
[Runtime files](#)
[Script links](#)
[Translations](#)
[Old stuff](#)

[Scripts](#)

Downloading Vim

Vim is available for many different systems and there are several versions. This page will help you decide what to download.

Most popular:

Click this link to download the [self-installing executable \(using ftp\)](#).

MS-Windows: Signed MS-Windows files are available on the [vim-win32-installer site \(gvim_8.2.0012_x86_signed.exe](#) is recommended)

See the [GitHub](#) page, or [Mercurial](#), if you prefer that.

Unix: There is also an [Aquimage](#) which is build daily and runs on many Linux systems.

Mac: See the [MacVim](#) project for a GUI version and [Hemsikrew](#) for a terminal version

Details and options for:

- [Unix](#)
- [PC: MS-DOS and MS-Windows](#)
- [Amiga](#)
- [OS/2](#)

Esc
명령 모드

vi / vim 단축키 모음

~ 대소문자 전환	! 외부 명령	@ 매크로 실행	# 이전 검색	\$ 줄 끝으로 이동	% 일치하는 괄호 찾기	^ 줄의 첫 글자	& :8 번복	* 다음 검색	(문장 시작) 문장 끝	_ 아래줄로 이동	+ 다음 줄
` 매크로 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 위로 검색	Y 줄단위 복사	U 줄단위 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 줄 삭제 후 편집모드	D 끝까지 삭제	F 뒤로 검색	G 파일 끝으로 이동	H 화면 상단	J 줄 합치기	K 다음 행	L 화면 하단	: ex 명령줄	" 레지스터 지정	열 이동	
a 덧붙이기	s 편집모드	d 삭제	f 한 문자 찾기	g 확장 명령	h ←	j ↓	k ↑	l →	; t/T/t/F 명령 반복	' 매크로 이동	\ 사용 안함	
Z 종료	X 백스페이스	C 줄 끝까지 바꾸기	V 줄단위 비주얼 모드	B 이전 WORD	N 이전 (찾기)	M 화면 가운데	< 내어쓰기	> 들여쓰기	? 찾기 (뒤로)			
Z 확장 명령	x 글자 삭제	c 바꾸기	v 비주얼 모드	b 이전 단어	n 다음 (찾기)	m 마크 설정	t/T/t/F , 역순 검색	. 명령 반복	/ 찾기			

동작

커서를 이동하거나, 연산자가 동작할 범위를 지정합니다.

명령

바로 동작하는 명령, 빨간색은 편집 모드로 변경됩니다.

연산자

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

화장

특별한 키 함수로, 추가적인 키 입력이 필요합니다.

q

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

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

words: quux{foo, bar, baz};
WORDS: quux{foo, bar, baz};

주요 명령행 명령 ('ex'):

:w (저장), :q (종료), :q! (지정하지 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: 블록비주얼 모드 (vim 전용)

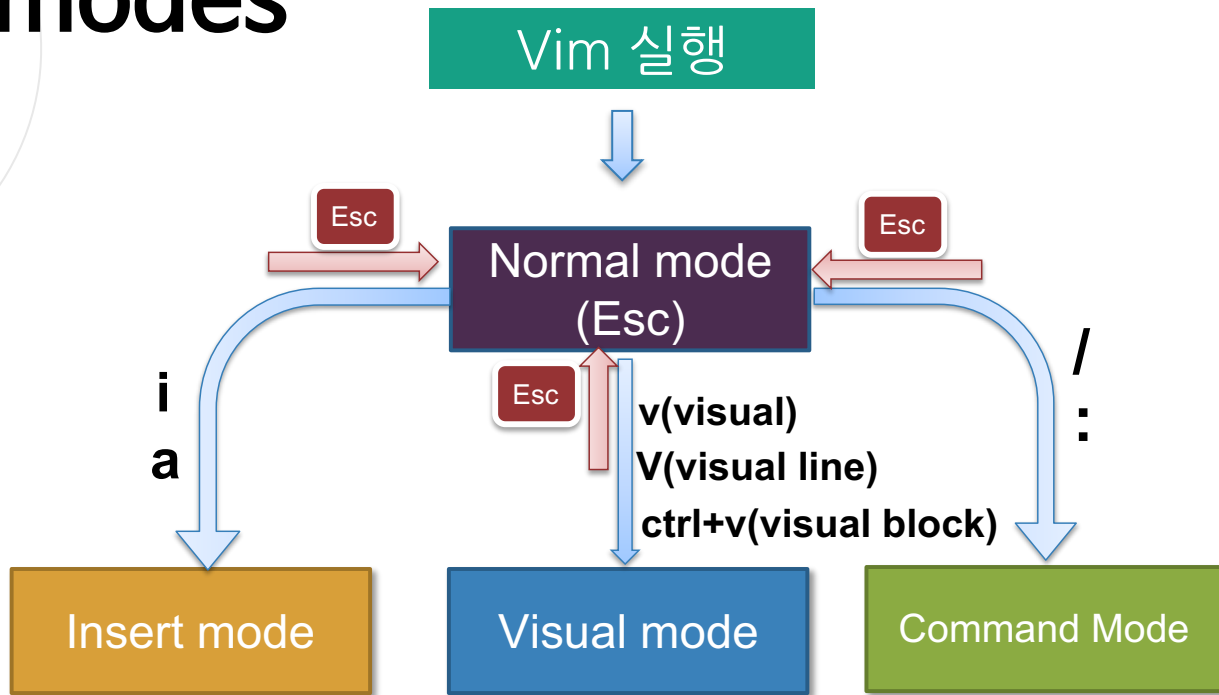
비주얼 모드:

커서를 움직여 지정한 범위에 연산자를 적용합니다. (vim 전용)

참고:

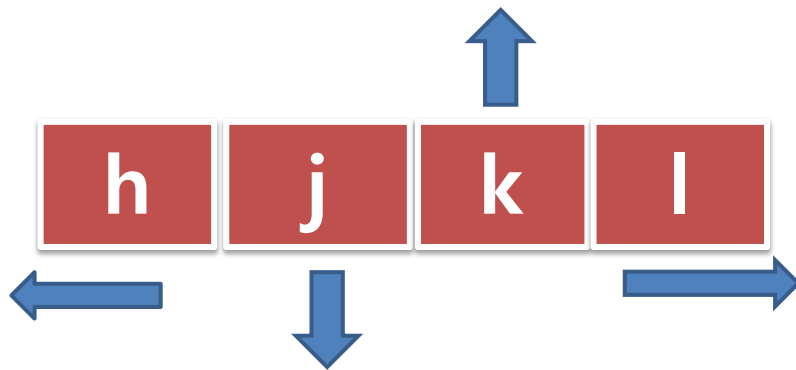
- (1) 복사/붙여넣기/지우기 명령어를 사용하기 전에 "x"를 입력하여 레지스터(편집모드)를 지정하세요. (x는 a에서 z 또는 * 을 사용할 수 있음) (예: "ay\$ 를 입력하면 현재 커서에서 라인 끝까지의 내용을 레지스터 'a'에 저장합니다.)
- (2) 어떤 명령을 입력하기 전에 횟수를 지정하면, 횟수만큼 반복하게 됩니다. (예: 2p, d2w, 5l, d4j)
- (3) 연속으로 입력하는 명령은 현재의 라인에 반영됩니다. 예시: dd(현재 라인 지우기), >>(들여쓰기)
- (4) ZZ 는 저장 후 종료, ZQ 는 저장하지 않고 종료.
- (5) zt : 커서가 위치한 곳을 제일위로 올리거나, zb : 바닥으로, zz : 가운데로
- (6) gg : 파일의 처음으로(Vim 전용), gf : 커서가 위치한 곳의 파일 열기(Vim 전용)

Vim modes

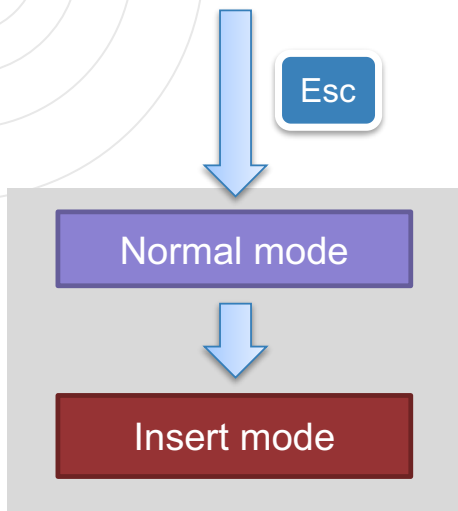


Vim cursor

- [ESC] 클릭 후 일반모드로 전환하여 사용



Vim - Insert mode



- [ESC] 클릭 후 일반모드로 전환하여 사용
- i 현재 커서 앞에서 편집 시작
- a 현재 커서 다음부터 편집 시작
- A 현재 커서 줄의 맨 끝으로 이동하고 편집
- o 현재 커서 아래 새로운 줄 추가하여 편집
- O 현재 커서 위 새로운 줄을 추가하고, 편집

Vim - save

- [ESC] 클릭 후 일반모드로 전환하여 사용
- **:wq!** save + quit
- **ZZ** save + quit
- **:q!** quit
- **:w newfile** save as

```
walab-HGU:~/20210SS/lab5:> vim test.c
```

```
walab-HGU:~/20210SS/lab5:> cat test.c
```

```
#include <stdio.h>
```

```
int main(){  
    printf("Hello!!!");  
    return 0;  
}
```

```
walab-HGU:~/20210SS/lab5:> vim test.c
```

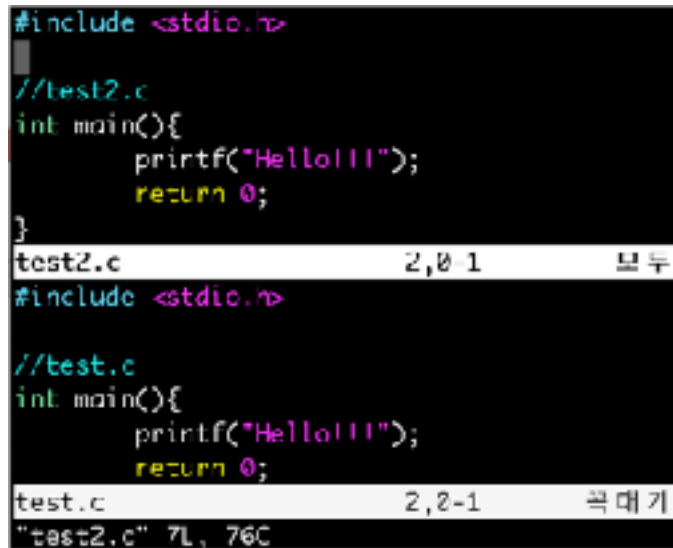
```
walab-HGU:~/20210SS/lab5:> ls
```

```
test.c  test2.c
```

test2.c 로 저장

Vim - open

- [ESC] 클릭 후 일반모드로 전환하여 사용
- 여러 파일을 열기
 - **:vs filename** 수직 창 나누고 읽어옴
 - **:split filename** 수평 창 나누고 읽어옴
 - **Ctrl + ww** 다른 창으로 이동
- 외부 명령어 실행하기
 - **:! command[enter]** 쉘 명령어 실행 후 Vim 복귀



```
#include <stdio.h>

//test2.c
int main(){
    printf("Hello!!!");
    return 0;
}

test2.c 2,0-1 모두

#include <stdio.h>

//test.c
int main(){
    printf("Hello!!!");
    return 0;
}

test.c 2,0-1 꼭대기
"test2.c" 7L, 76C
```

Vim - copy / paste

- [ESC] 클릭 후 일반모드로 전환하여 사용
- yy 현재 라인 복사
- 3yy 현재 커서에서 3라인 복사
- p 현재 커서 아래 붙여넣기
- P 현재 커서 위에 붙여넣기

```
#include <stdio.h>

int main(){
    printf("Hello World!!!\n");
    printf("\n");
    return 0;
}
```

```
#include <stdio.h>

int main(){
    printf("Hello World!!!\n");
    printf("Hello World!!!\n");
    printf("\n");
    return 0;
}
```

Vim - delete / undo / redo

- [ESC] 클릭 후 일반모드로 전환하여 사용
- **x** 한 글자 지우기
- **dw** 현재 커서에 있는 한 단어 지우기
- **dd** 현재 라인 지우기
- **3dd** 현재 커서부터 3 라인 지우기
- **u** 마지막 명령 취소
- **U** 해당 줄 전체의 수정사항 취소
- **Ctrl + r** redo 기능

```
#include <stdio.h>

int main(){
    printf("lo World!!!\n");
    printf("Hello World!!!\n");
    printf("\n");
    return 0;
}
```



```
#include <stdio.h>

int main(){
    printf("Hello World!!!\n");
    printf("\n");
    return 0;
}
```

Vim - replace / move

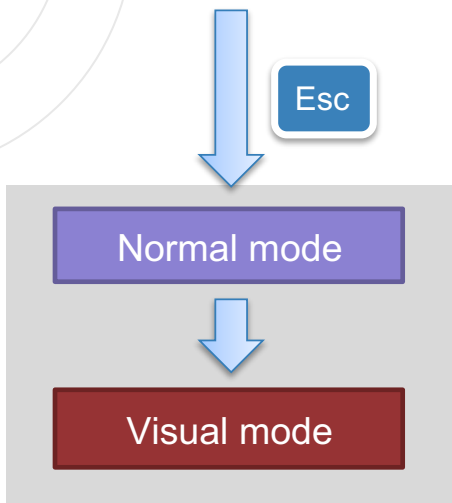
- [ESC] 클릭 후 일반모드로 전환하여 사용
- **r** 한 글자 수정하기
- **cw** 단어를 변경할 때 사용(삭제후 입력)
- **c\$** 해당 줄 전체를 변경할 때 사용(삭제 후 입력)

- **E** 현재 라인의 끝으로 이동(\$ 도 동일)
- **G** 파일의 끝으로 이동
- **gg** 파일의 처음으로 이동
- **3G** 3번째 라인으로 이동

Vim – find / search /replace

- [ESC] 클릭 후 일반모드로 전환하여 사용
- /keyword 원하는 keyword 검색
- ?keyword 원하는 keyword를 역방향으로 검색
- n 검색 결과에서 다음 문자열 찾기
- N 검색 결과에서 이전 문자열 찾기
- :%s/old/new/g 찾는 단어를 새 단어로 파일에서 전체(globally) 변경
- :%s/old/new/gc 사용자에게 물으며 변경
- :%s/old/new/i 대소문자 구분 없이 검색
- :#,#s/old/new/g 줄 번호(#) 사이의 찾는 단어를 새 단어로 모두 변경

Vim - visual mode



- [ESC] 클릭 후 일반모드로 전환하여 사용
- **v** 현재 커서위치부터 문자단위로 선택 가능
- **V** 현재 커서위치부터 라인 단위로 선택 가능
- **Ctrl + v** 현재 커서를 기준으로 블록 선택 가능
- **p** 복사된 데이터를 현재 위치에 붙여넣기
- **P** 복사된 데이터를 마지막 라인에 붙여넣기

과제설명

1. The Linux Command Line의 chapter6~ chapter10에 설명되어 있는 각 명령어를 아래 Step을 참조하여 실습한 후, 정리하세요. (웹에서 검색해서 이해해도 좋습니다)

교재 pdf : [Download LinuxCommand](#)

Steps

- walab.handong.edu에 접속하여 각 명령어를 실행
- 어떤 기능을 수행하는 명령어인지 본인이 이해하고 설명 작성
- 각 명령이 실행결과 화면 캡처

* 단, 명령어 중 권한이 없어서 결과가 나오지 않는 경우에는 화면캡처하여 제출

* 단 chap9, chap10은 명령어에 대한 설명만 제출

2. 아래 실습자료를 번호별로 walab서비에 접속하여 실습해보고, 각 번호에 필요한 정보를 각 번호별로 정리하세요.

정리가 필요한 항목 : 각 번호에 필요한 것만

- a. shell 명령어
- b. test.c 파일 변경내용
- c. vim으로 수행하기 위해 필요한 키보드 키

실습자료:

=====

* 홈디렉터리는 본인 홈디렉터리를 의미함

* vim 으로 작업을 하다가 영방이 되면 저장하지 않고 종료(q!)하며 새로 작성하거나 취소(일반모드 u) 할 것

1. 홈디렉터리에 2021OSS디렉터리 생성 및 이동
2. 2021OSS디렉터리내에 lab5 디렉터리 생성 이동
3. 홈디렉터리에 lab5 디렉터리를 비어 갈 수 있는 symbolic link 생성(이름 : oss_lab5)

MINI Project

2. CRUD구현

Walab 서버에 접속해서 코딩할 것!

* 제출내용 : product.c 소스 + mini_project 폴더에서 ls -al 실행결과 및 프로그램 실행 각 함수별 테스트 결과화면 캡처

* 제출시 이름_product.pdf 한 파일에 저장하여 제출할 것

지난 과제에서 설계한 product.h을 토대로 CRUD(Create/Read/Update/Delete) 기능의 함수를 코딩하고, 각 함수를 테스트 해 보세요. (단, 하나의 데이터(하나의 과일)만을 처리하도록 함수를 코딩)



[베리타운] 청블루베리 210g
6,980원
(100g당 2,252원)
★★★★★ (3,719명)



[베리타운] Delic 스페치오 바나나 1.3kg 내
지
4,980원
★★★★★ (22,193명)



[육산의 집] 푸드계 망고요리요 900g/팩
9,900원
(100g당 1,100원)
★★★★★ (3,219명)



[베리타운] 사과 4~11월/종 1.3kg
12,800원
(100g당 712원)
★★★★★ (4,283명)

1. walab.handong.edu 서버에 접속
2. 홈 디렉터리 아래 2021OSS/mini_project/ 폴더 생성
3. product.h 파일 생성 및 저장(copy & paste)
4. product.c 파일 생성 및 CRUD 함수 코딩
5. product.c 파일에 각 함수 테스트를 위한 main함수 코딩(각 함수가 정상적으로 작동하는지 테스트할 것)
6. 컴파일 및 실행