

Using VI Editor

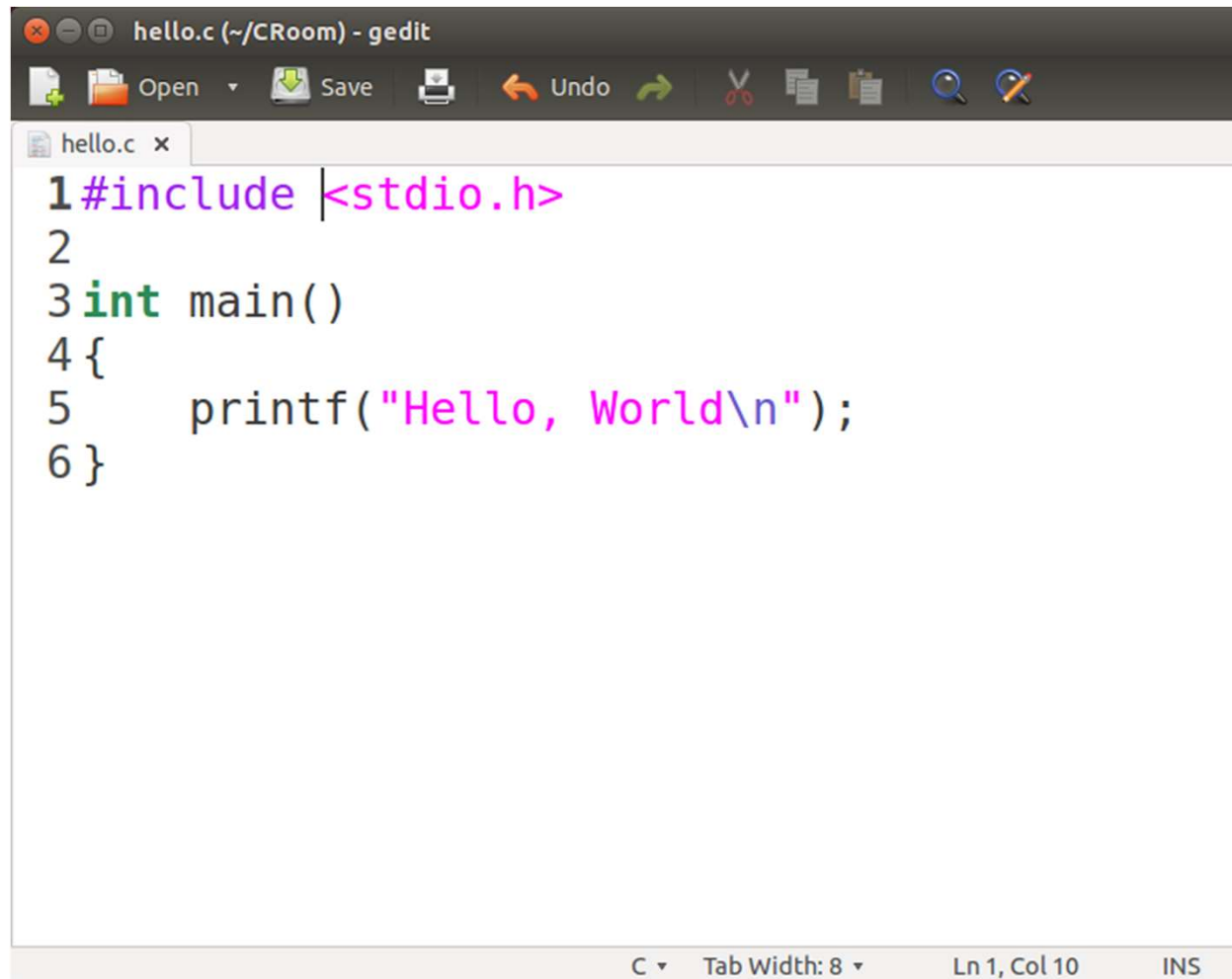


Bok, Jong Soon

javaexpert@nate.com

<https://github.com/swacademy/Ubuntu>

Linux Text Editors - gedit



The image shows a screenshot of the gedit text editor window. The title bar at the top reads "hello.c (~/CRoom) - gedit". Below the title bar is a menu bar with "Open", "Save", "Undo", and other icons. The main editing area contains the following C code:

```
1#include <stdio.h>
2
3int main()
4{
5    printf("Hello, World\n");
6}
```

The status bar at the bottom indicates "C", "Tab Width: 8", "Ln 1, Col 10", and "INS".

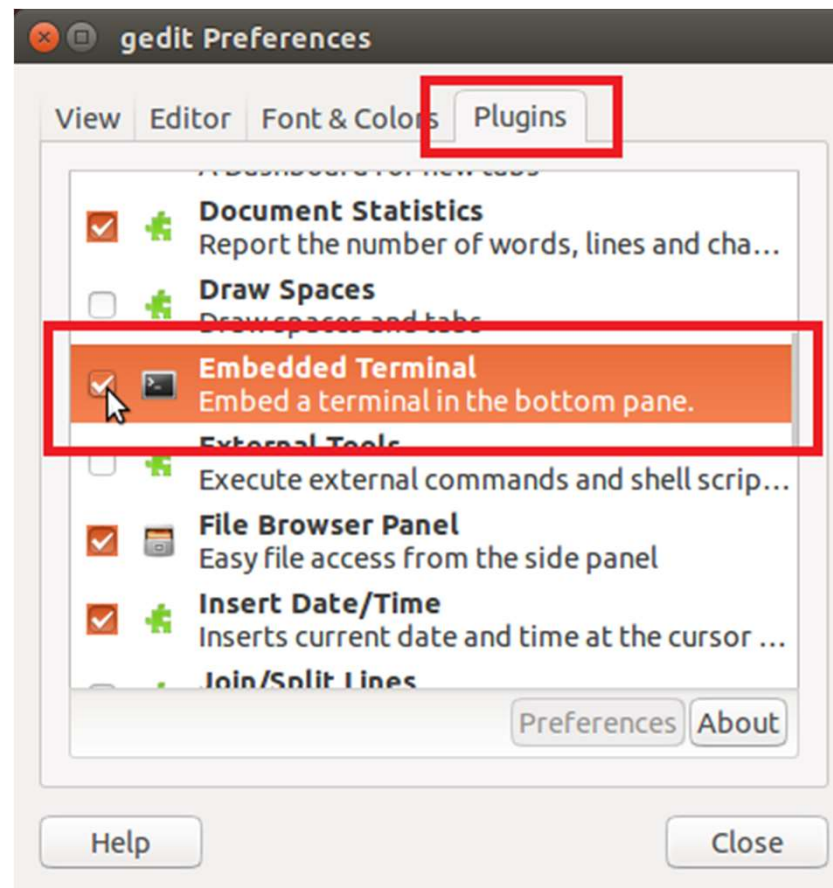
Lab 1 : Installation gedit Plugins

Installation gedit Plugins (1/3)

1. `$ sudo apt-get install -y gedit-plugins`

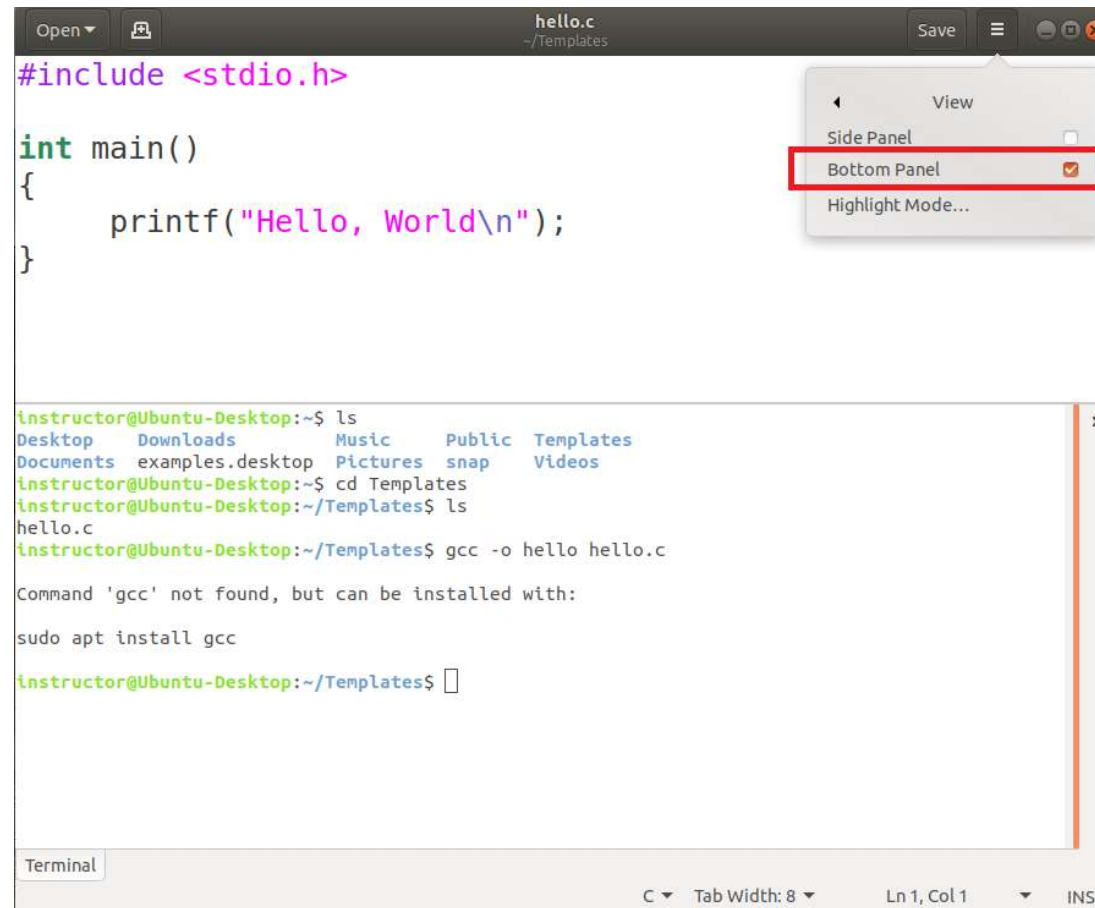
Installation gedit Plugins (2/3)

2. Preferences > Plugins > Embedded Terminal

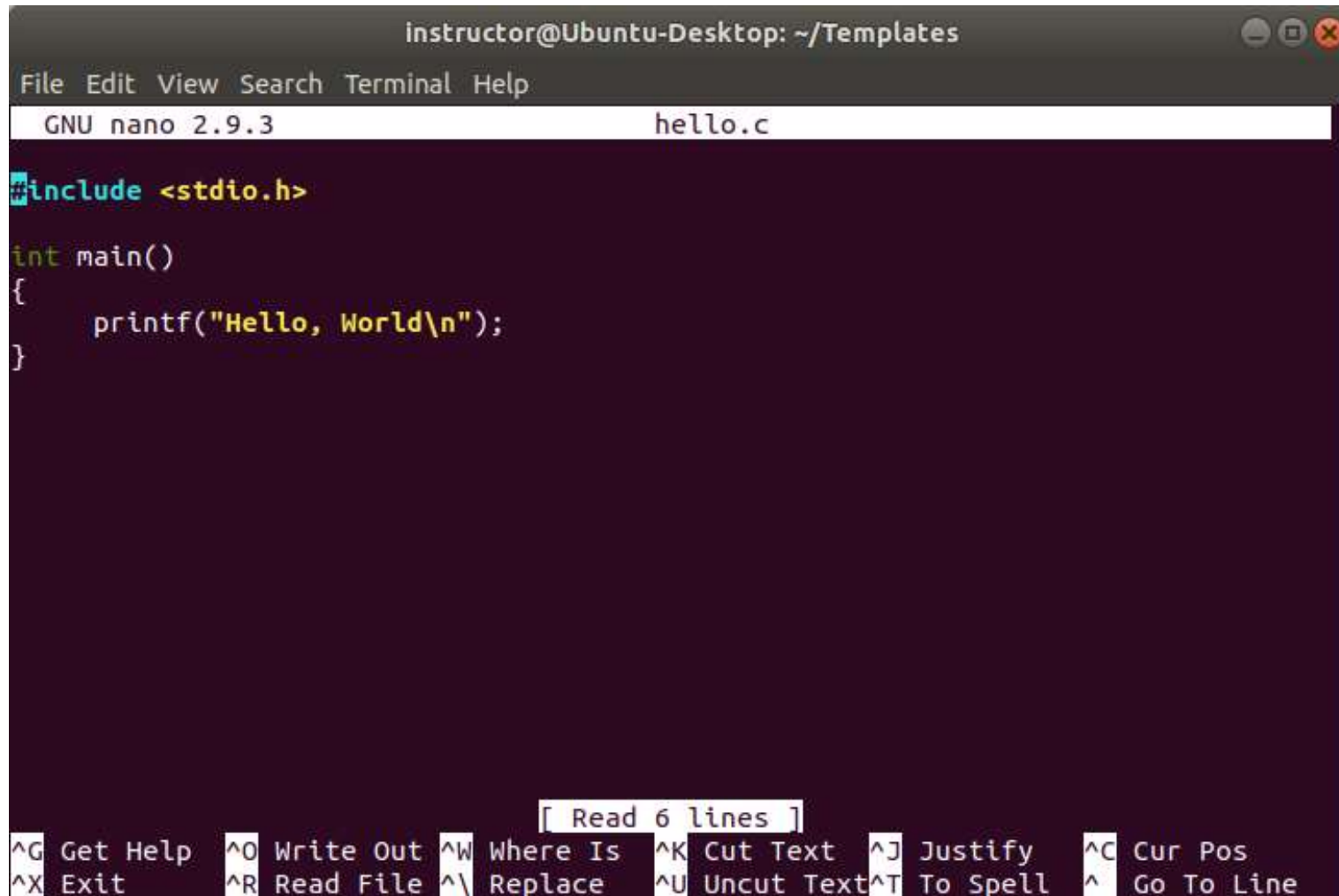


Installation gedit Plugins (3/3)

3. Check View > Bottom Panel



Linux Text Editors – Nano



The screenshot shows the GNU nano 2.9.3 text editor running in a terminal window. The window title is "instructor@Ubuntu-Desktop: ~/Templates". The editor is editing a file named "hello.c". The code in the file is:

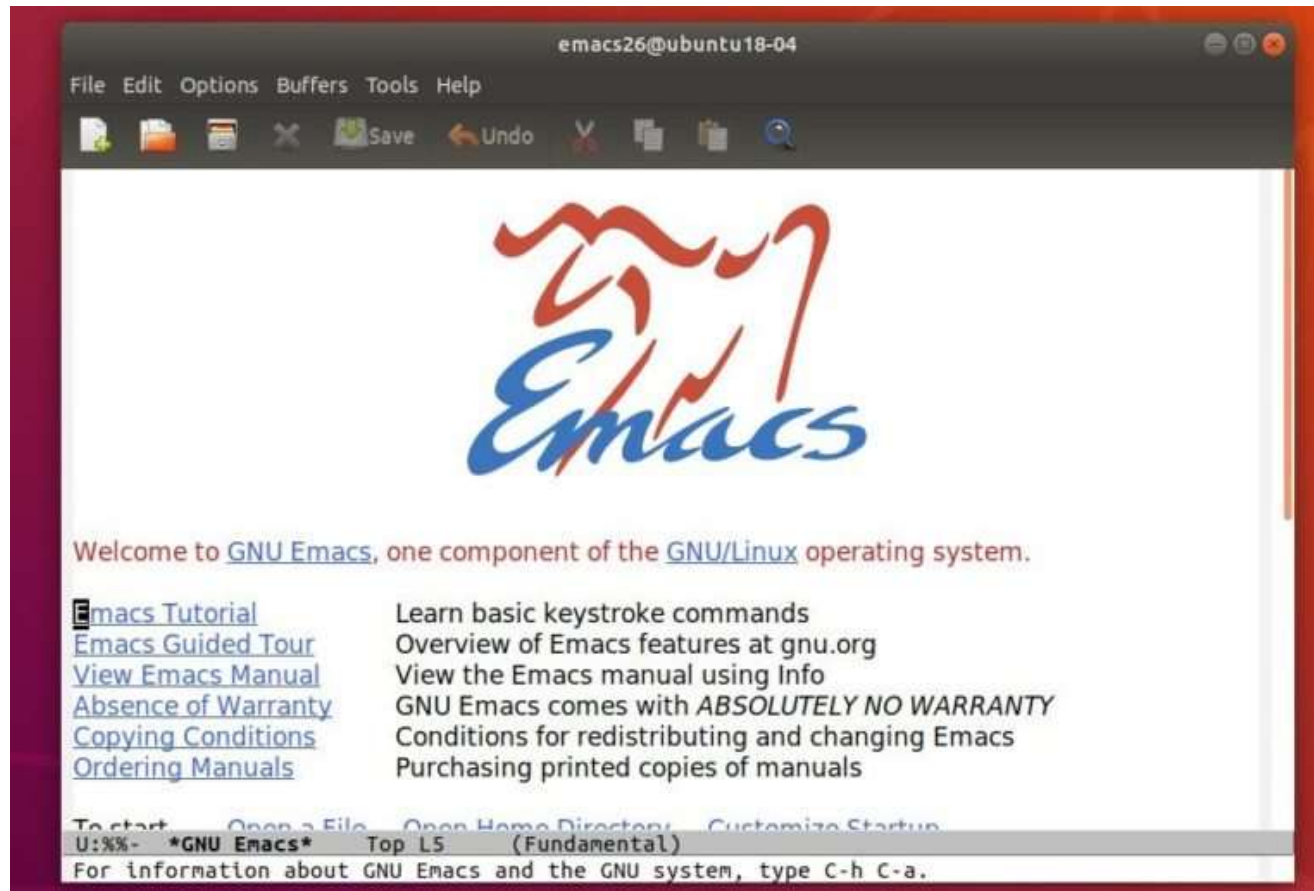
```
include <stdio.h>

int main()
{
    printf("Hello, World\n");
}
```

The bottom status bar shows the following information:

- [Read 6 lines]
- ^G Get Help
- ^O Write Out
- ^W Where Is
- ^K Cut Text
- ^J Justify
- ^C Cur Pos
- ^X Exit
- ^R Read File
- ^_ Replace
- ^U Uncut Text
- ^T To Spell
- ^_ Go To Line

Linux Text Editors - Emacs



Lab 2 : Installation Emacs on Ubuntu 18.04 LTS

Installation Emacs (1/5)

1. `$ sudo add-apt-repository ppa:kelleeyk/emacs`

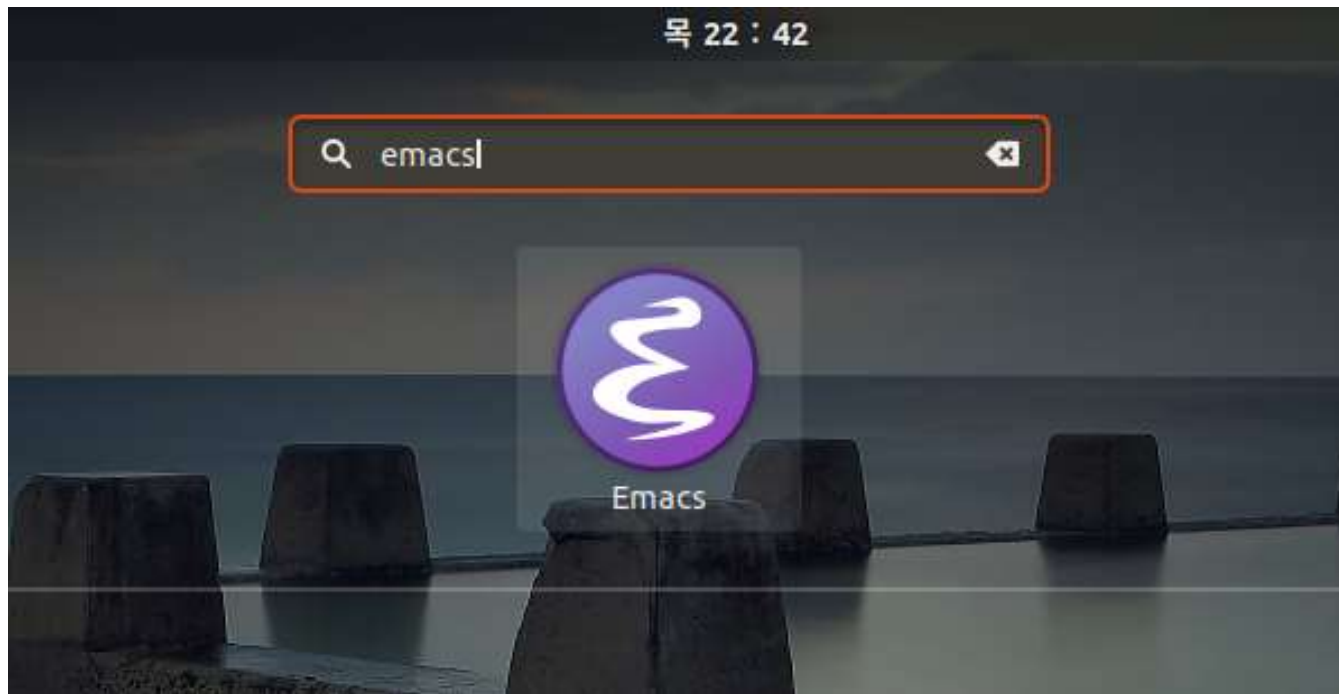
Installation Emacs (2/5)

2. `$ sudo apt-get update`

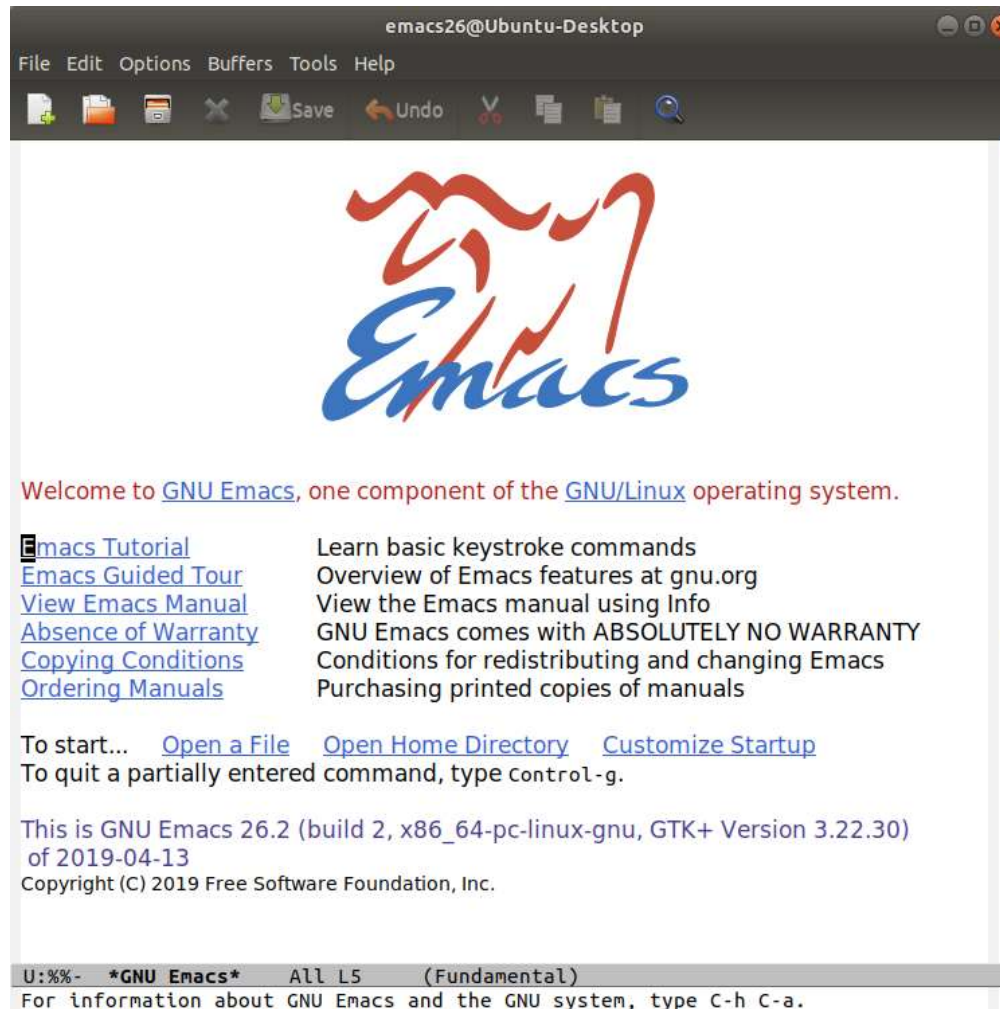
Installation Emacs (3/5)

3. `$ sudo apt-get install emacs26`

Installation Emacs (4/5)



Installation Emacs (5/5)



Linux Text Editors – VI or VIM

[illegible]

Why use **VI** Editor?

- Although easy-to-use graphical text editors, most power users still use **VI** to edit text files.
- **VI** will work from any shell (no GUI required).
- Offer other advantages such as your hands never having to leave the keyboard and integration with useful utilities.
- Are usable over slow Internet connections such as dial-up or satellite.

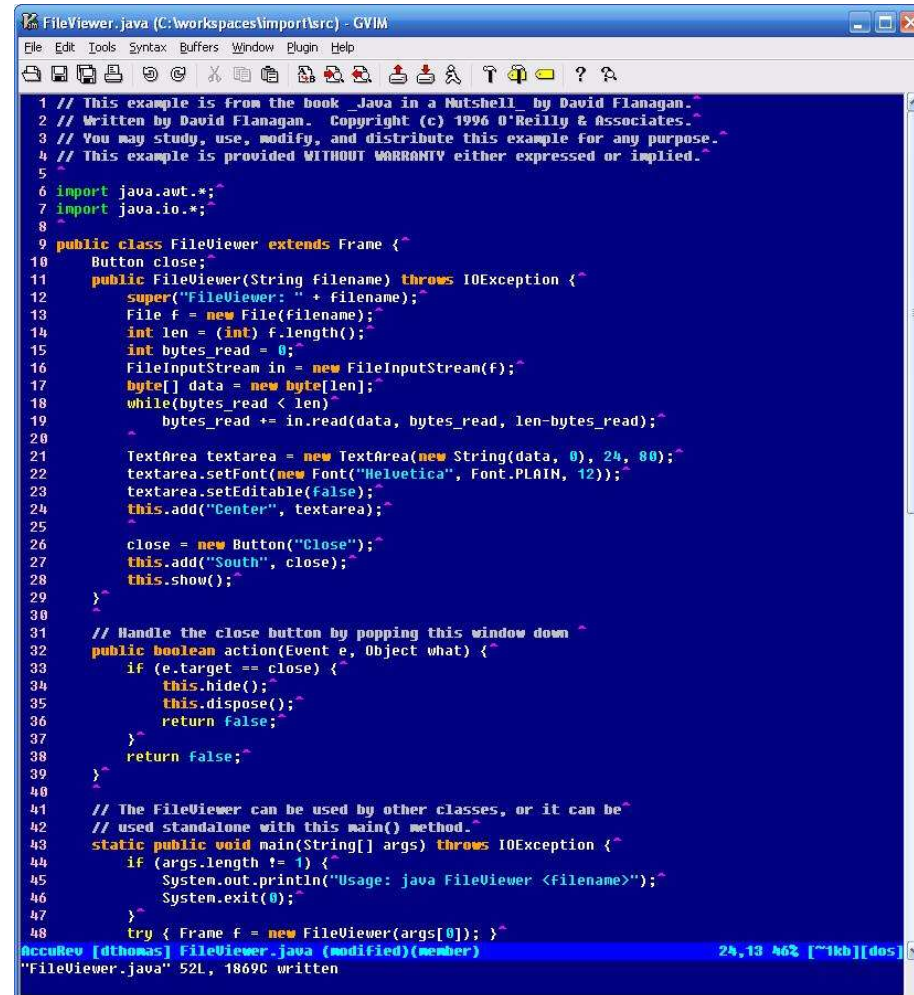
What is **vi** ?

- Is pronounced */ˈviːaɪ/*
- Is a screen-oriented text editor originally created for the Unix.
- The original code was written by Bill Joy, Chuck Haley in 1976.
- The visual mode for a line editor called **ex**.
- Bill Joy's **ex** 1.1 was released as part of the first BSD Unix release in March, 1978.



Why learning VI ?

- Occasions
- Operating Systems
- Efficiency
- Regular Expression
- Quick-lunch keys
- License
- Etc.

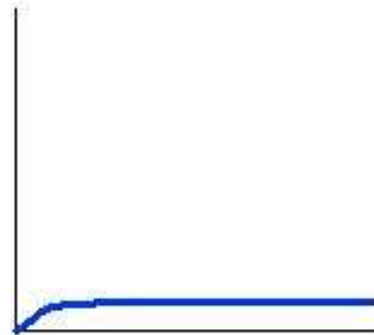


```
1 // This example is from the book _Java in a Nutshell_ by David Flanagan.
2 // Written by David Flanagan. Copyright (c) 1996 O'Reilly & Associates.
3 // You may study, use, modify, and distribute this example for any purpose.
4 // This example is provided WITHOUT WARRANTY either expressed or implied.
5
6 import java.awt.*;
7 import java.io.*;
8
9 public class FileViewer extends Frame {
10     Button close;
11     public FileViewer(String filename) throws IOException {
12         super("FileViewer: " + filename);
13         File f = new File(filename);
14         int len = (int) f.length();
15         int bytes_read = 0;
16         FileInputStream in = new FileInputStream(f);
17         byte[] data = new byte[len];
18         while(bytes_read < len)
19             bytes_read += in.read(data, bytes_read, len-bytes_read);
20
21         TextArea textarea = new TextArea(new String(data, 0), 24, 80);
22         textarea.setFont(new Font("Helvetica", Font.PLAIN, 12));
23         textarea.setEditable(false);
24         this.add("Center", textarea);
25
26         close = new Button("Close");
27         this.add("South", close);
28         this.show();
29     }
30
31     // Handle the close button by popping this window down
32     public boolean action(Event e, Object what) {
33         if (e.target == close) {
34             this.hide();
35             this.dispose();
36             return false;
37         }
38         return false;
39     }
40
41     // The FileViewer can be used by other classes, or it can be
42     // used standalone with this main() method.
43     static public void main(String[] args) throws IOException {
44         if (args.length != 1) {
45             System.out.println("Usage: java FileViewer <Filename>");
46             System.exit(0);
47         }
48         try { Frame f = new FileViewer(args[0]); }
49     }
50 }
```

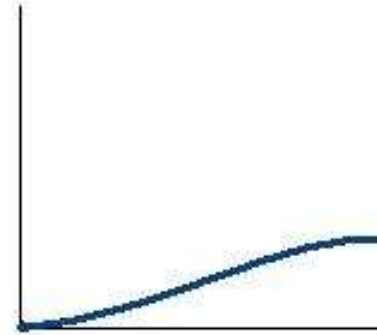
Learning curves

Classical learning
curves for some
common editors

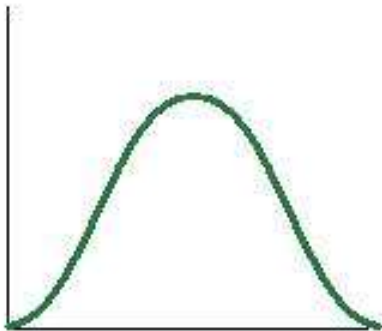
Notepad



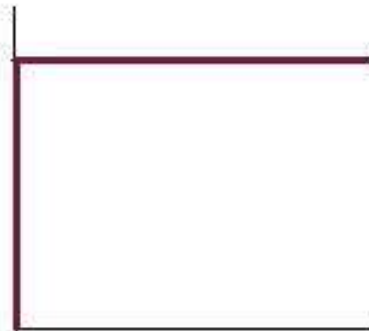
Pico



Visual Studio



vi



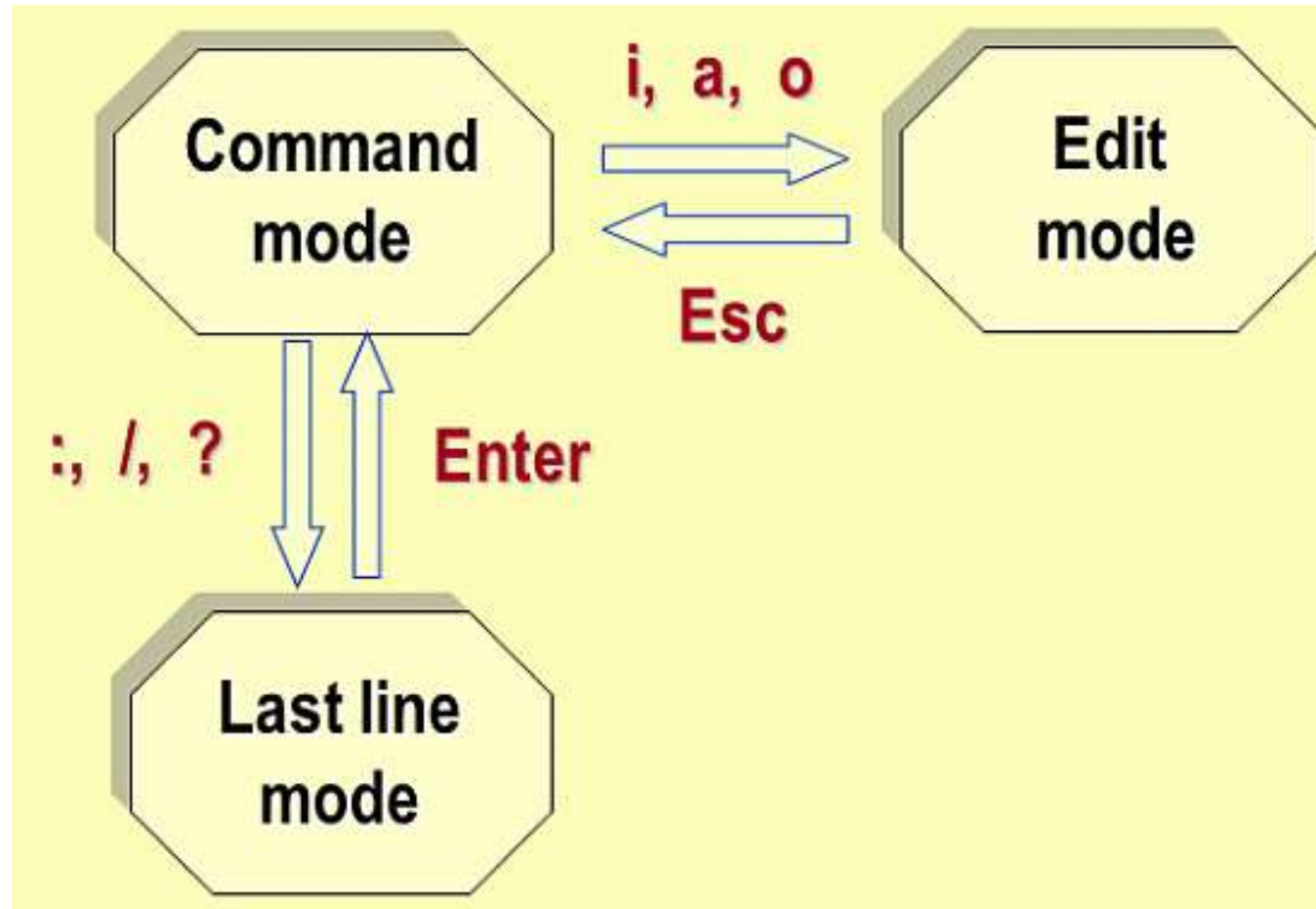
emacs



Features

- Powerful text editor
- Character based
- Interactive
- Cryptic
- Available on virtually ALL Unix systems
- Designed to work with a alphanumeric only terminal (no arrow keys, pgup, pgdw, etc)
- The most letters with less number of keystrokes.
- Syntax highlight for most languages.
- All natural languages.
- Thousands of plug-ins.
- Automatic backup (.swp)
- Highly customizable
- Without mouse

VI 3 Modes



VI 3 Modes (Cont.)

■ Six modes

NAME	DESCRIPTION	HELP PAGE
normal	Navigation & Editing text	:help Normal-mode
insert	Inserting new text	:help Insert-mode
visual	Navigation & Manipulating selections	:help visual-mode
select	Similar to visual (but like MS-Window)	:help select-mode
command-line mode	Entering editor commands	:help Command-line-mode
Ex-mode	Similar to command-line (but optimized for batch processing)	:help Ex-mode

VI Simple Cheat Sheet

version 1.1
April 1st, 06

vi / vim graphical cheat sheet

Esc
normal mode

~ toggle case	! external filter	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* next ident	(begin sentence) end sentence	"soft" bol down	+ next line
\ goto mark	1	2	3	4	5	6	7	8	9	0 "hard" bol	- prev line	= auto format

Q ex mode	W next WORD	E end WORD	R replace mode	T back 'till	Y yank line	U undo line	I insert at bol	O open above	P paste before	{ begin parag.	} end parag.
q record macro	w next word	e end word	r replace char	t 'till	y yank	u undo	i insert mode	o open below	p paste after	[misc] misc

A append at eol	S subst line	D delete to eol	F "back" find ch	G eof/ goto ln	H screen top	J join lines	K help	L screen bottom	. ex cmd line	! reg. spec	bol/ goto col
a append	s subst char	d delete	f find char	g extra cmds	h ←	j ↓	k ↑	l →	; repeat t/T/f/F	' goto mk. bol	\ not used!

Z quit	X back-space	C change to eol	V visual lines	B prev WORD	N prev (find)	M screen mid'l	< un-indent	> indent	? find (rev.)
Z extra cmds	x delete char	c change	v visual mode	b prev word	n next (find)	m set mark	, reverse t/T/f/F	. repeat cmd	/ find

Legend:

- motion** moves the cursor, or defines the range for an operator
- command** direct action command, if red, it enters insert mode
- operator** requires a motion afterwards, operates between cursor & destination
- extra** special functions, requires extra input

q. commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line, mk = mark, yank = copy

words: quux(foo, bar, baz);
WORDS: quux(foo, bar, baz);

Main command line commands ('ex'):

- :w (save), :q (quit), :q! (quit w/o saving)
- :e f (open file f),
- :%s/x/y/g (replace 'x' by 'y' filewide),
- :h (help in vim), :new (new file in vim),

Other important commands:

- CTRL-R: redo (vim),
- CTRL-F/-B: page up/down,
- CTRL-E/-Y: scroll line up/down,
- CTRL-V: block-visual mode (vim only)

Visual mode:

Move around and type operator to act on selected region (vim only)

Notes:

- (1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay\$ to copy rest of line to reg 'a')
- (2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)
- (3) duplicate operator to act on current line (dd = delete line, >> = indent line)
- (4) ZZ to save & quit, ZQ to quit w/o saving
- (5) zt: scroll cursor to top, zb: bottom, zz: center
- (6) gg: top of file (vim only), gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

VI Simple Cheat Sheet (Cont.)

Legend:

- Macro** Register name (0-9a-zA-Z) required
- Op** Motion req.; act between cursor & dst
- Cmd** Command
- Ins** Command and enter insert mode
- Move** Moves cursor or defines range for op
- Find** Search (↖ = reverse, ↗ = forward)
- Tag** ctags / diffs / folding
- Code** Code formatting, whitespace, etc.
- Extra** Extended functionality; req. extra chars
- Char arg req.** g z Z w ' " ~

word `Foo { [src , b dst , b len] ;`

WORD `Foo (src , b dst , b len) ;`

Startup

```
vim <filename> +123      goto line 123
vim <file> -t Foo         edit at tag 'Foo'
vim <file> -c "Foo"       cmd: find 'Foo' & edit
gvim -g or gvim          start GUI ver.
```

Linux `:set guifont=ProgygTinyT12`

OSX `:set guifont=ProgygTiny:h11`

diff `vimdiff <file1> <file2> [<file3>]`

Broken Keys Ctrl-I = Tab, Ctrl-I = ESC

Vim is still unable to map certain keys for your own use...

See: `arc/ops.c -c "f valid_yank_req" for * reg. names`

See: `arc/norm.c -c "fiv_cmds" for g* extra cmds`

See: `arc/edit.c -c "ctrl_x_mags" for * insert cmds`

0 `See: ops.c, Ctrl-I, Ctrl-Shift-I, Ctrl-I, Ctrl-I, etc.`

1 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

2 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

3 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

4 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

5 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

6 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

7 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

8 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

9 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

0 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

1 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

2 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

3 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

4 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

5 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

6 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

7 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

8 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

9 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

0 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

1 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

2 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

3 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

4 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

5 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

6 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

7 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

8 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

9 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

0 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

1 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

2 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

3 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

4 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

5 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

6 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

7 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

8 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

9 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

0 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

1 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

2 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

3 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

4 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

5 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

6 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

7 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

8 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

9 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

0 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

1 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

2 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

3 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

4 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

5 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

6 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

7 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

8 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

9 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

0 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

1 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

2 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

3 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

4 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

5 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

6 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

7 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

8 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

9 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

0 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

1 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

2 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

3 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

4 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

5 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

6 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

7 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

8 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

9 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

0 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

1 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

2 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

3 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

4 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

5 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

6 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

7 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

8 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

9 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

0 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

1 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

2 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

3 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

4 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

5 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

6 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

7 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

8 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

9 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

0 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

1 `See: arc/edit.c -c "ctrl_x_mags" for * insert cmds`

vi Simple Cheat Sheet (Cont.)

version 1.1
April 1st, 06

vi / vim 단축키 모음

Esc
명령 모드

~ 대소문자 전환	! 외부 명령	@ 매크로 실행	# 이전 검색	\$ 줄 끝으로 이동	% 일치하는 곳으로 찾기	^ 줄의 첫 글자	& :s 반복	* 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	T 뒤로 검색	Y 줄단위 복사	U 줄단위 실행 취소	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 실행 취소	i 편집 모드	o 행 아래에 삽입	P 커서 이후에 붙여넣기	[기타]	기타
A 줄 끝에 덧붙이기	S 줄 삭제 후 편집 모드	D 줄 끝까지 삭제	F 뒤로 검색	G 파일 끝/줄로 이동	H 화면 상단	J 줄 합치기	K 도움말	L 화면 하단	: ex 명령줄	" 레지스터 지정	열 이동	
a 덧붙이기	s 단어 삭제 후 편집 모드	d 1,3 삭제	f 한 문자 찾기	g 확장 명령	h ←	j ↓	k ↑	l →	. t/T/I/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 1,3 바꾸기	v 비주얼 모드	b 이전 단어	n 다음 (찾기)	m 마크 설정	t/T/I/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 사용해야 하며, 커서의 위치부터 목적지까지 연산합니다.

확장 특별한 키 합수로, 추가적인 키 입력이 필요합니다.

q 입력 후 (숫자를 제외한 .으로 끝낼 수 있는) 글자를 입력해야 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! (저장하지 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: 블록-비주얼 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(클립보드)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 횡수를 지정하면, 횡수만큼 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, ZQ는 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)

vi/vim 에 대한 더 많은 강좌나 팁을 얻으려면 www.viemu.com (ViEmu, MS 비주얼 스튜디오를 위한 vi/vim 에뮬레이션)을 방문하십시오.

VIM Installation

- <http://vim.sourceforge.net/download.php>
- <ftp://ftp.vim.org/pub/vim/MIRRORS>

VIM Installation (Cont.)

- `$ sudo apt-get install -y vim`

VIM Installation (Cont.)

The image displays two terminal windows from the Ubuntu Desktop environment. The top window, titled 'instructor@Ubuntu-Desktop: ~/Templates', shows the initial steps of creating a file named 'hello.c'. It starts with the command 'touch hello.c' followed by 'cat hello.c', which outputs the file's contents: '#include <stdio.h>', 'int main()', '{', 'printf("Hello, World\\n");', '}', and several tilde characters representing newlines. The bottom window, also titled 'instructor@Ubuntu-Desktop: ~/Templates', shows the same file being edited with a text editor. The code is identical to the one in the top window. At the bottom of this window, a status bar indicates '"hello.c" 6L, 66C' and '1,1 All', suggesting the file is 6 lines long and 66 characters wide, with the cursor at line 1, column 1.

Using the Vim tutor

- Can use the *vimtutor* to learn your first Vim commands.
- On Linux/Unix, if *Vim* has been properly installed, can start it from the shell:

```
$ vimtutor
```


Using the Vim tutor (Cont.)

```
instructor@Ubuntu-Desktop: ~/Templates
File Edit View Search Terminal Help
=====
=  Welcome to the VIM Tutor - Version 1.7  =
=====

Vim is a very powerful editor that has many commands, too many to
explain in a tutor such as this. This tutor is designed to describe
enough of the commands that you will be able to easily use Vim as
an all-purpose editor.

The approximate time required to complete the tutor is 25-30 minutes,
depending upon how much time is spent with experimentation.

ATTENTION:
The commands in the lessons will modify the text. Make a copy of this
file to practice on (if you started "vimtutor" this is already a copy).

It is important to remember that this tutor is set up to teach by
use. That means that you need to execute the commands to learn them
properly. If you only read the text, you will forget the commands!

Now, make sure that your Shift-Lock key is NOT depressed and press
the  j  key enough times to move the cursor so that Lesson 1.1
completely fills the screen.
"/tmp/tutorqfmKba" 970 lines, 33248 characters
```

Starting the VI

- **VI** is working on a copy of your file in the *buffer*.
- Will *not* affect original file until save the *buffer*.

```
$ vi [filename]
```

- If the filename is omitted, **VI** will open an unnamed buffer.
- A filename
 - Must be unique inside its directory.
 - Can include any ASCII character except a slash.

Starting the **VI** (Cont.)

1. \$ **cp** /usr/share/doc/base-files/copyright /tmp
2. \$ **vi** /tmp/copyright

- Begin on line 10

\$ **vi** **+10** [filename]

Starting the **vi** (Cont.)

- Begin editing file on the last line

```
$ vi + [filename]
```

- Edit file in read-only mode

```
$ view [filename]
```

```
$ vi -R [filename]
```

- To display line numbers

```
:set nu Return
```

Saving & Exiting the VI

- First check that you are in command mode by pressing [ESC]
- To save file and quit the VI:
 - ZZ
 - :wq
 - :x Return
- To save file without quitting VI:
 - :w <filename> Return (write to specified file)
 - :w Return (write again)
 - :w! <filename> Return (to override existing file)
 - :w >> filename Return
- To quit without saving changes:
 - :q! Return (Exit – DON'T save changes)

Basic Editing

version 1.1
April 1st, 06

vi/vim lesson 1 - basic editing

motion

moves the cursor, or defines the range for an operator

command

direct action command, if **red**, it enters insert mode

Esc

normal mode

\$

eol

^

"soft" bol

0

"hard" bol

W

next WORD

E

end WORD

R

replace mode

w

next word

e

end word

u

undo

i

insert mode

A

append at eol

h

←

j

↓

k

↑

l

→

.

ex cmd line

X

back-space

x

delete char

B

prev WORD

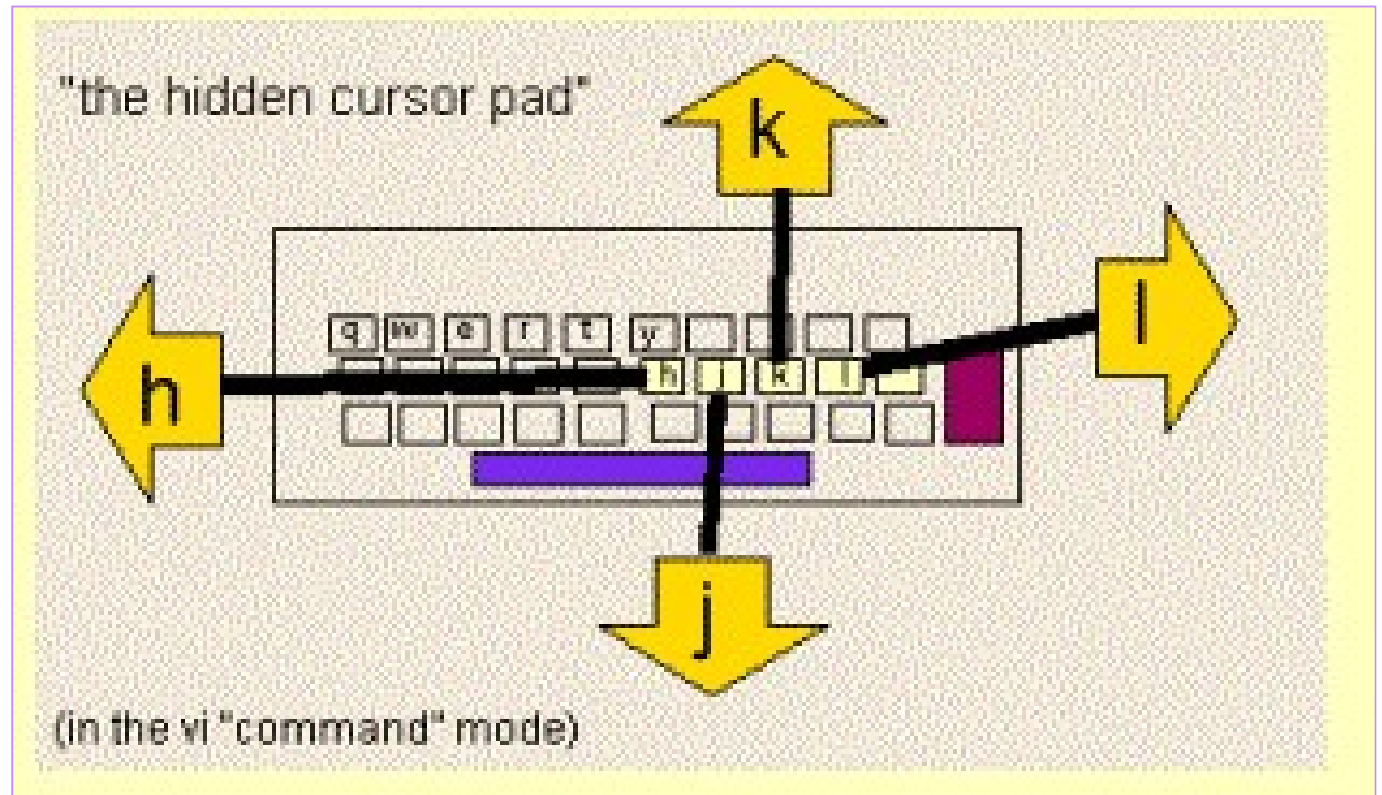
b

prev word

Basics:
h j k l are vi/vim cursor keys – use them as they are much closer than regular cursor keys!
Use **i** to enter insert mode, cursor turns from a block into a vertical line, and you can type in text. Use **Esc** to return to normal mode.
Use **x** to delete the current character, or **X** to delete the one to the left
Use **A** to go insert text at the end of the line (wherever you are in the line!)
*(Note: insert mode is actually very similar to a regular editor, you can use cursor/navigation keys, backspace, delete...)***Extras:**
u to undo the last action – traditional vi has a single level, while vim supports unlimited undo (CTRL - **R** to redo)
0 jumps directly to the beginning of the line, **\$** to the end, and **^** to the first non-blank
Use **w b e** to move along 'words'. A 'word' is a sequence of all alphanumeric or punctuation signs: **quux(foo, bar, baz);**
Use **W B E** to move along WORDS. A 'WORD' is a sequence of any non-blank characters: **quux(foo, bar, baz);**
Use **R** to enter insert mode with an overstrike cursor, which types over existing characters.
: **w** and press enter to save, **:** **q** and enter to quit.For the rest of the tutorial & a full cheat sheet, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

Moving the Cursor

- Up **k**
 - Down **j**
 - Left **h**
 - Right **l**
-
- N <direction>
 - **3h**
 - **5k**



Moving the Cursor (Cont.)

Command	Description
0 or 	Moves to <i>beginning</i> of the current line.
^	Moves to the first letter of the current line
\$	Moves to the <i>end</i> of the current line.
-(minus)	Moves to beginning of <i>previous</i> line.
+(plus)	Moves to beginning of <i>next</i> line
w	Move cursor to <i>next</i> word.
b	Move cursor to <i>previous</i> word.
e	Move to the end of your word.
: \$	Move to the <i>last</i> line.
G	Move to the <i>last</i> line.
gg	Move to the <i>first</i> line.
1G	Move to the <i>first</i> line.

Moving the Cursor (Cont.)

Command	Description
(Moves to <i>beginning</i> of sentence
)	Moves to <i>end</i> of sentence
{	Moves to <i>beginning</i> of paragraph
}	Moves to <i>end</i> of paragraph
H	Moves to <i>top</i> of screen(window)
M	Moves to <i>middle</i> of screen(window)
L	Moves to <i>bottom</i> of screen(window)
: <i>n</i>	Moves to the line number
<i>n</i> G	Moves to the line number

Control Commands

Command	Description
Ctrl + <i>d</i>	Moves forward ½ screen full
Ctrl + <i>u</i>	Moves backward ½ screen full
Ctrl + <i>f</i>	Moves forward 1 screen full
Ctrl + <i>b</i>	Moves backward 1 screen full
Ctrl + <i>e</i>	Moves screen up one line
Ctrl + <i>y</i>	Moves screen down one line
Ctrl + <i>l</i>	Refresh screen (useful in telnet)

Entering insert mode in **VI**

Command	Description
i	Insert text <i>before</i> cursor
I	Insert text at <i>beginning</i> of current line
a	Append text <i>after</i> cursor
A	Append text at <i>end</i> of current line
o	Open new line <i>above</i> current line
O	Open new line <i>below</i> current line

Deleting Characters in VI

Command	Description
x	Deletes the character <i>under</i> the cursor location.
X	Deletes the character <i>before</i> the cursor location.
dw	Deletes to the <i>next</i> word.
d^	Deletes to the <i>beginning</i> of the line.
d0	Deletes to the <i>beginning</i> of the line.
d\$	Deletes to the <i>end</i> of the line.
D	Deletes to the <i>end</i> of the current line.
dd	Deletes the line the cursor is on.

Change Commands (Insert mode) in **VI**

Command	Description
cc	Change line, blanks line. To clean, ESC.
cw	Change word to the \$ sign. To clean, ESC.
c\$	Change to end of line. To clean, ESC.
C	Change to end of line. To clean, ESC.
r	Replace current character
R	Overwrite until ESC
s	Substitute (ESC) – 1 char with string
S	Substitute (ESC) – Rest of line with text

Copy and Paste Commands in VI

Command	Description
yy	Copies the current line.
yw	Copies the current word.
p (lower)	Puts the copied text <i>after</i> the cursor.
P (Upper)	Puts the copied text <i>before</i> the cursor.

Using Visual Block Mode in **VI**

Command	Description
v	Start visual mode per character.
V	Start visual mode linewise.
Ctrl+V	Start visual mode blockwise.
y	Copy
~	Switch case
d	Delete
dd	Delete current line
p	Puts
c	Change

Searching in VI

Command		Description
<i>/chars</i>	<input type="text" value="Return"/>	Search forward for <i>chars</i>
<i>?chars</i>	<input type="text" value="Return"/>	Search backward for <i>chars</i>
N		Repeat search in <i>opposite</i> direction
n		Repeat search in <i>same</i> direction
/	<input type="text" value="Return"/>	Repeat search <i>forward</i>
?	<input type="text" value="Return"/>	Repeat search <i>backward</i>

Regular Expressions (Search Strings)

Character	Description
^	Matches beginning of line
.	Matches any single character
*	Matches any previous character
\$	Matches end of line
. *	Matches any character

Finding help

- To get generic help use
 - `:help`
- To get help on a given subject
 - `:help subject`
 - Ex) `:help intro.txt`
- To get help on the “x” command
 - `:help x`
- To find out how to delete
 - `:help deleting`

Miscellaneous

Commands	Description
J	Join the current line with the next one.
<<	Shifts the current line to the left by one shift width.
>>	Shifts the current line to the right by one shift width.
~	Switch the case of the character under the cursor.
U	Undo all changes on line
u	Undo last change
Ctrl+u	Redo
.	Repeat last change

Miscellaneous (Cont.)

Commands	Description
: !command	Runs command from editor
:r filename	Reads file and inserts it after current line.
:nr filename	Reads file and inserts it after line <i>n</i> .
:e filename	Opens another file with filename.
:f	Displays current position in the file in % and filename, total number of file.
:f filename	Renames current file to filename.

VIM Tabs

- To open many files in tabs
 - \$ `vi -p *.txt`
- To switch between tabs
 - Type `gt`
- To create a new empty tab
 - `:tabnew`
- To open a file in a new tab
 - `:tabe filename`

VIM Tabs (Cont.)

A screenshot of a terminal window titled "instructor@Ubuntu-Desktop: ~/Templates". The terminal shows a vi editor interface editing a file named "vi_exercise.dat". The file contains a simple C program:

```
1 #include <stdio.h>  
2  
3 int main()  
4 {  
5     printf("Hello, World\n");  
6 }
```

The first three tabs in the editor are labeled "hello1.c", "hello.c", and "vi_exercise.dat". A red rectangular box highlights the tab "vi_exercise.dat". At the bottom right of the terminal, the cursor position "1,1" and the word "All" are visible.

VIM Window

- To split the current window horizontally
 - `:split(:sp)` or `Ctrl + w, s`
- To split the current window vertically
 - `:vertical split(:vs)` or `Ctrl + w, v`
- To switch between open windows
 - `Ctrl + w, {h, j, k, l}`
- To close the current window
 - `Ctrl + w, c`
- To close all windows except the current one.
 - `Ctrl + w, o`

VIM Window (Cont.)

The screenshot shows a terminal window titled 'instructor@Ubuntu-Desktop: ~/Templates'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The main area is split vertically into two panes, both displaying the same C code:

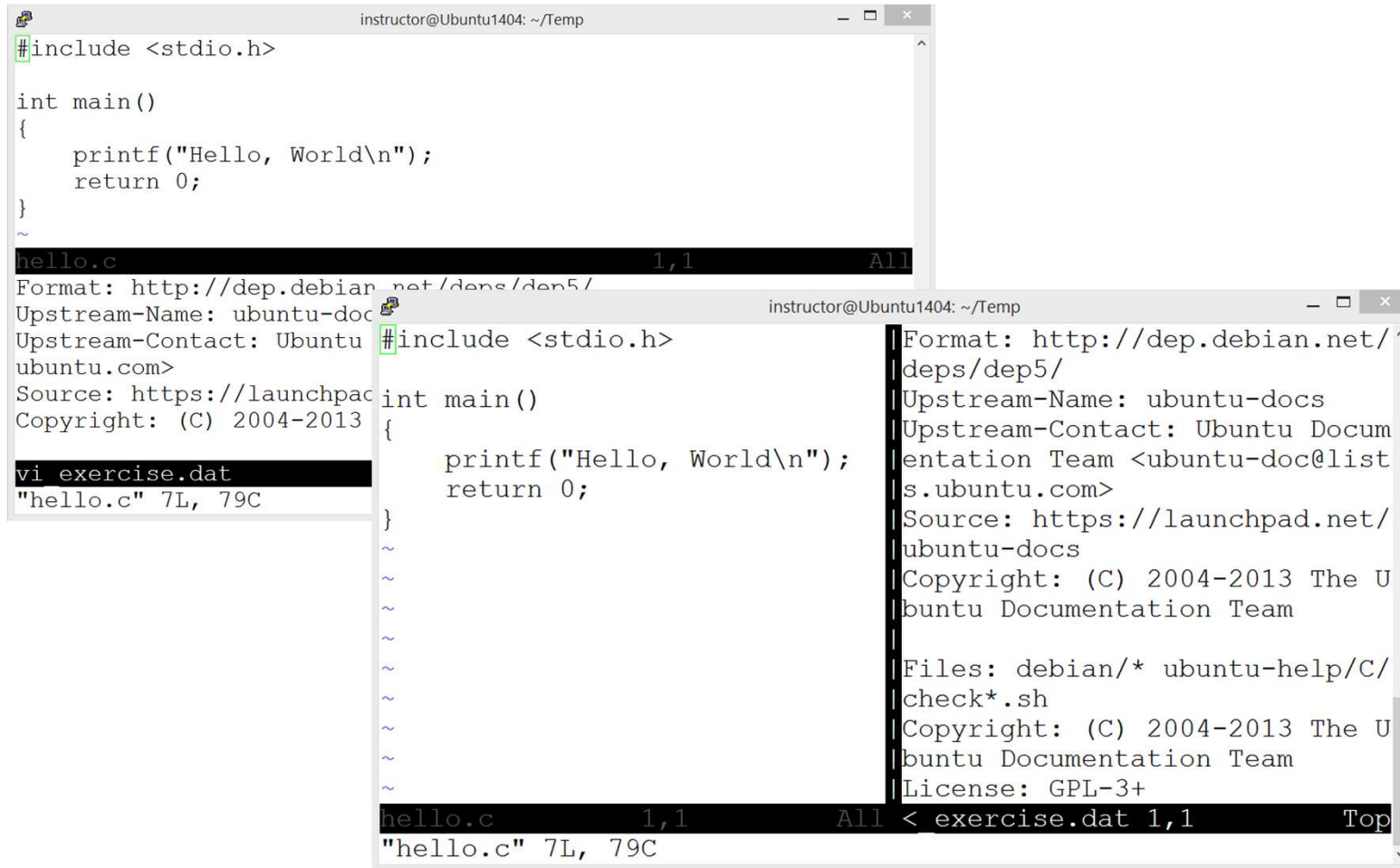
```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello, World\n");
6 }
```

At the bottom, the status bar shows 'hello.c' and 'vertical split'.

VIM Window (Cont.)

- To open a new file
 - `:new` or `Ctrl + w, n`
- To open a file horizontally
 - `:sp filename`
- To open a file vertically
 - `:vsp filename`
- To switch between windows in command mode.
 - `Ctrl + w {hjkl}`

VIM Window (Cont.)



```
instructor@Ubuntu1404: ~/Temp
#include <stdio.h>

int main()
{
    printf("Hello, World\n");
    return 0;
}
~
hello.c 1,1 All
```

```
instructor@Ubuntu1404: ~/Temp
Format: http://dep.debian.net/deps/dep5/
Upstream-Name: ubuntu-docs
Upstream-Contact: Ubuntu Documentation Team <ubuntu-doc@lists.ubuntu.com>
Source: https://launchpad.net/ubuntu-docs
Copyright: (C) 2004-2013 The Ubuntu Documentation Team

Files: debian/* ubuntu-help/C/check*.sh
Copyright: (C) 2004-2013 The Ubuntu Documentation Team
License: GPL-3+

hello.c 1,1 All < exercise.dat 1,1 Top
"hello.c" 7L, 79C
```

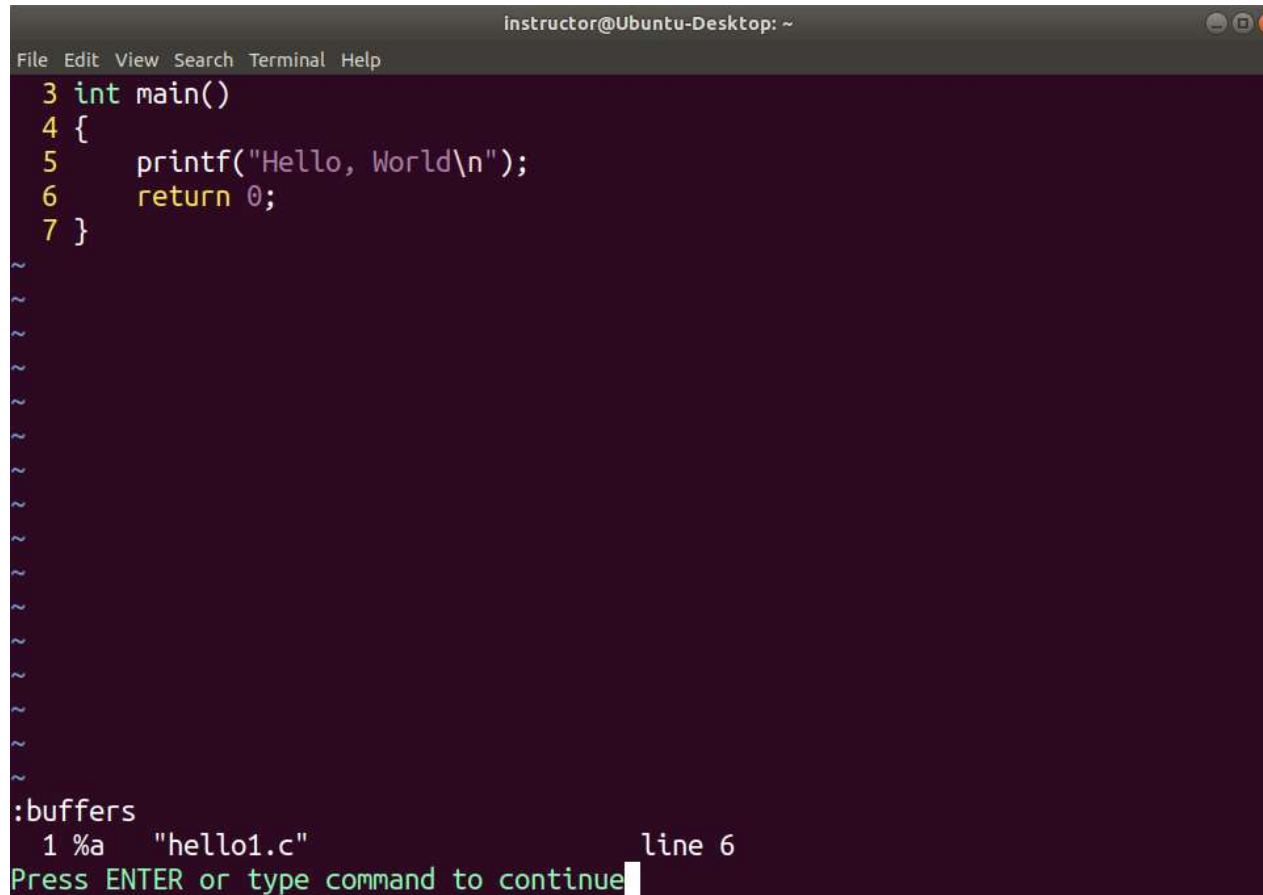

VIM Window (Cont.)

- To maximize in current working window horizontally
 - `Ctrl + w, _ (underline)`
- To maximize in current working window vertically
 - `Ctrl + w, | (pipe sign)`
- To divide equal size
 - `Ctrl + w, =`

VIM Buffer

- List of open buffers
 - `:ls`
- Previous buffer
 - `:bp` or `:N`
- Next buffer
 - `:bn` or `:n`
- Move to n'th buffer
 - `:b NUMBER`
- To save current open file
 - `:w`

VIM Buffer (Cont.)



The screenshot shows a VIM editor window titled "instructor@Ubuntu-Desktop: ~". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The editor contains a C program with the following code:

```
3 int main()
4 {
5     printf("Hello, World\n");
6     return 0;
7 }
```

Below the code, there are several tilde (~) characters representing buffer names. At the bottom, the command `:buffers` has been executed, showing the output:

```
1 %a "hello1.c" line 6
```

Below the output, the text "Press ENTER or type command to continue" is displayed.

Using shell commands

- To escape temporarily to a shell
 - `:sh`
- To return from shell to VI
 - `Ctrl + d`
- To execute shell command without leaving VI
 - `:! command`

Set Commands

Command	Description
<code>:set ic</code> or <code>:set ignorecase</code>	Ignores case when searching
<code>:set ai</code> or <code>:set autoindent</code>	Sets auto indent
<code>:set noai</code> or <code>:set noautoindent</code>	To unset auto indent.
<code>:set nu</code> or <code>:set number</code>	Show line numbers
<code>:set nonu</code> or <code>:set nonumber</code>	Don't show line numbers
<code>:set sm</code> or <code>:set showmode</code>	Display mode on last line of screen
<code>:set nosm</code> or <code>:set noshowmode</code>	Turn off show mode
<code>:set ts=n</code> or <code>:set tabstop=n</code>	Sets the width of a software tabstop

Set Commands (Cont.)

- To display current values of VI variables
 - **:set all**

```
instructor@Ubuntu-Desktop: ~  
File Edit View Search Terminal Help  
:set all  
--- Options ---  
aleph=224          noexec          modelines=5        statusline=  
noarabic           fileencoding=    modifiable        suffixesadd=  
arabicshape       fileformat=unix  modified          swapfile  
noallowrevins     nofileignorecase more              swapsync=fsync  
noaltkeymap       filetype=        mouse=           switchbuf=  
ambiwidth=single  fixendofline    mousemodel=extend synmaxcol=3000  
noautochdir       nofkmap         mousetime=500     syntax=  
noautoindent      foldclose=      number           tabline=  
noautoread        foldcolumn=0    numberwidth=4     tabpagemax=10  
noautowrite       foldenable      omnifunc=         tabstop=8  
noautowriteall    foldexpr=0      operatorfunc=     tagbsearch  
background=dark   foldignore=#    nopaste          tagcase=followic  
nobackup          foldlevel=0     pastetoggle=      taglength=0  
backupcopy=auto   foldlevelstart=-1 patchexpr=        tagrelative  
backupext=~       foldmethod=manual patchmode=        tagstack  
backupskip=/tmp/* foldminlines=1   nopreserveindent notermbidi  
balloondelay=600  foldnestmax=20  previewheight=12  termencoding=  
noballoonvalterm  formatexpr=     nopreviewwindow  notermguicolors  
balloonexpr=      formatoptions=tcq printdevice=      termkey=  
belloff=         formatprg=      printencoding=    termsize=  
nobinary         fsync           printfont=courier noterse  
-- More --
```

Using vimrc

- Contains optional runtime configuration settings to initialize Vim when it starts.
- On Linux/Unix based systems, the file is named *.vimrc*.
- Locate in *\$HOME*.
- Examples → *:help vimrc-intro*

```
" Always wrap long lines:  
set wrap  
" Show line number :  
set number
```

Lab 3 : Practice with VIM

Practice with VIM

1. `$ cp /usr/share/doc/ubuntu-docs/copyright vi_exercise.dat`
2. `$ vi vi_exercise.dat`

Practice with VIM (Cont.)

```
instructor@Ubuntu-Desktop: ~/Templates
File Edit View Search Terminal Help
Format: http://dep.debian.net/deps/dep5/
Upstream-Name: ubuntu-docs
Upstream-Contact: Ubuntu Documentation Team <ubuntu-doc@lists.ubuntu.com>
Source: https://launchpad.net/ubuntu-docs
Copyright: (C) 2004-2014 The Ubuntu Documentation Team

Files: *
Copyright: (C) 2004-2014 The Ubuntu Documentation Team
License: GPL-3+
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

.
You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>.

.
On Debian Systems, the full text of this license can be found in
"vi_exercise.dat" 319L, 22462C                                     1,1      Top
```

VIM Reference

■ Tutorial

- <http://vimdoc.sourceforge.net/>
- <https://danielmiessler.com/study/vim/>
- <http://www.openvim.com/>
- <https://linuxconfig.org/vim-tutorial>
- https://blog.interlinked.org/tutorials/vim_tutorial.html
- <https://scotch.io/tutorials/getting-started-with-vim-an-interactive-guide>

■ VIM Learning Game

- <http://vim-adventures.com/>