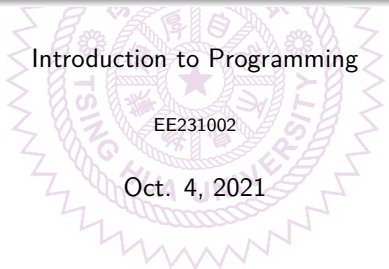


# Introduction to Vim, I

Introduction to Programming

EE231002

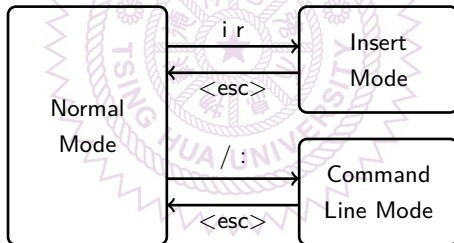
Oct. 4, 2021





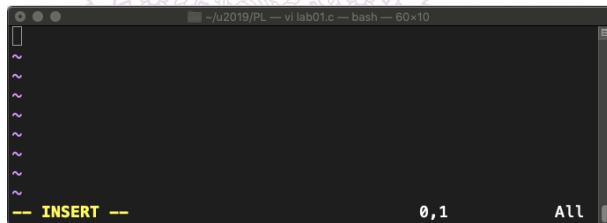
# Three Modes in vim

- There are three modes in **vim**
  - Normal mode: copy, delete, paste
  - Insert mode: insert text
  - Command line mode: save file, exit, search and replace



# Inserting Text

- When `vim` starts, it enters normal mode
- Press `i` to enter insert mode
  - Note the `-- INSERT --` on the lower-left corner
  - You can type in C program at this time



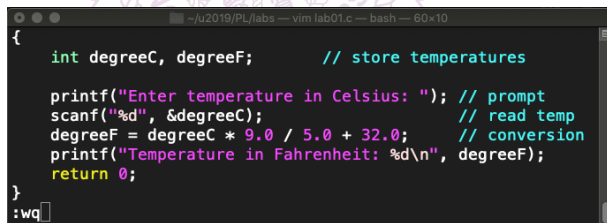
The screenshot shows a terminal window with a dark background. The title bar at the top reads `~/u2019/PL — vi lab01.c — bash — 60x10`. The main area of the terminal is mostly empty, with a cursor at the top left. On the left side, there are several tilde characters (~) indicating line numbers. At the bottom left, the text `-- INSERT --` is displayed in yellow. At the bottom right, the text `0,1` and `All` are visible, indicating the current cursor position and the extent of the insert mode.

# Insert Mode

- In insert mode, you can type in texts
- To move cursor
  - `↑`, `↓`, `←`, `→` keys move cursor in four directions
  - `PgUp` and `PgDn` keys scroll one page of text
  - `Home` key moves cursor to the beginning of the line
  - `End` key moves cursor to the end of the line
  - `Tab` key moves cursor to fixed columns (4x or 8x)
    - In our labs please use `Tab` key for indentation and each `Tab` key moves 4 spaces
- Press `Esc` key to return to normal mode

# Quitting vim

- In normal mode, the following commands save file or quit `vim` program
  - `:w`: save typed inputs to the file
  - `:q`: quit `vim` program (no saving file)
  - `:q!`: forced quitting from `vim` program
    - Changes are not updated to the file
  - `:wq`: save file and then quit `vim` program
  - `ZZ`: same as `:wq` but is a normal mode command
- Note that that the above except `ZZ` are executed in command line mode



```
~/u2019/PL/labs — vim lab01.c — bash — 60x10
{
    int degreeC, degreeF;        // store temperatures

    printf("Enter temperature in Celsius: "); // prompt
    scanf("%d", &degreeC);           // read temp
    degreeF = degreeC * 9.0 / 5.0 + 32.0; // conversion
    printf("Temperature in Fahrenheit: %d\n", degreeF);
    return 0;
}
:wq
```

# Show Line Numbers in vim

- `vim` does not show line numbers by default
  - Line numbers are very useful in debugging compiler errors
  - To show line number type in `:set nu` in normal mode

```
~/u2019/PL/labs — vim lab01.c — bash — 63x10
{
    int degreeC, degreeF;        // store temperatures

    printf("Enter temperature in Celsius: "); // prompt
    scanf("%d", &degreeC);           // read temp
    degreeF = degreeC * 9.0 / 5.0 + 32.0; // conversion
    printf("Temperature in Fahrenheit: %d\n", degreeF);
    return 0;
}
:set nonu                               13,10-13      Bot
```

```
~/u2019/PL/labs — vim lab01.c — bash — 63x10
6 {
7     int degreeC, degreeF;        // store temperatures
8
9     printf("Enter temperature in Celsius: "); // prompt
10    scanf("%d", &degreeC);           // read temp
11    degreeF = degreeC * 9.0 / 5.0 + 32.0; // conversion
12    printf("Temperature in Fahrenheit: %d\n", degreeF);
13    return 0;
14 }
:set nu
```

# Color Text

- `vim` takes advantage of the color terminal to make the file more legible
- The text color can be turned off by using `:syntax off` command
- `:syntax on` turns on color text

The image shows two screenshots of a vim editor window. The top screenshot shows the code with syntax highlighting: keywords are green, comments are cyan, strings are magenta, and the return statement is yellow. The status bar at the bottom of the window shows `:syntax on` on the left and `13,10-13 Bot` on the right. The bottom screenshot shows the same code but with no syntax highlighting. The status bar at the bottom of this window shows `:syntax off` on the left and `13,10-13 Bot` on the right.

```
{
    int degreeC, degreeF;        // store temperatures

    printf("Enter temperature in Celsius: "); // prompt
    scanf("%d", &degreeC);           // read temp
    degreeF = degreeC * 9.0 / 5.0 + 32.0; // conversion
    printf("Temperature in Fahrenheit: %d\n", degreeF);
    return 0;
}
:syntax on                                13,10-13    Bot
```

```
{
    int degreeC, degreeF;        // store temperatures

    printf("Enter temperature in Celsius: "); // prompt
    scanf("%d", &degreeC);           // read temp
    degreeF = degreeC * 9.0 / 5.0 + 32.0; // conversion
    printf("Temperature in Fahrenheit: %d\n", degreeF);
    return 0;
}
:syntax off                               13,10-13    Bot
```

- This is the mode that I use to view your program
  - Be sure your program is very legible to me in this mode



## Color Text, II

- Depending on terminal background, the text color may need to be adjusted

- `:set bg=dark`

```
~/u2019/PL/labs — vim lab01.c — bash — 63x10
{
    int degreeC, degreeF;        // store temperatures

    printf("Enter temperature in Celsius: "); // prompt
    scanf("%d", &degreeC);           // read temp
    degreeF = degreeC * 9.0 / 5.0 + 32.0; // conversion
    printf("Temperature in Fahrenheit: %d\n", degreeF);
    return 0;
}
:set bg=dark
```

- `:set bg=light`

```
~/u2019/PL/labs — vim lab01.c — bash — 63x10
{
    int degreeC, degreeF;        // store temperatures

    printf("Enter temperature in Celsius: "); // prompt
    scanf("%d", &degreeC);           // read temp
    degreeF = degreeC * 9.0 / 5.0 + 32.0; // conversion
    printf("Temperature in Fahrenheit: %d\n", degreeF);
    return 0;
}
:set bg=light
```

# Auto-indent

- In insert mode, after typing a line of text the cursor moves to the first column – not aligned with the indented text
- This can be changed by `:set ai`, auto-indent, command
- `:set noai` sets no auto-indent

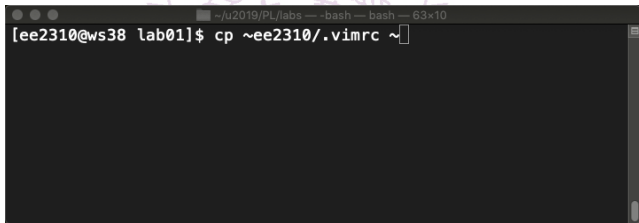
```
~/u2019/PL/labs — vim lab01.c — bash — 63x10
{
  int degreeC, degreeF;      // store temperatures

  printf("Enter temperature in Celsius: "); // prompt
  scanf("%d", &degreeC);           // read temp
  degreeF = degreeC * 9.0 / 5.0 + 32.0; // conversion
  printf("Temperature in Fahrenheit: %d\n", degreeF);
}
~
-- INSERT --                               13,1           Bot
```

```
~/u2019/PL/labs — vim lab01.c — bash — 63x10
{
  int degreeC, degreeF;      // store temperatures

  printf("Enter temperature in Celsius: "); // prompt
  scanf("%d", &degreeC);           // read temp
  degreeF = degreeC * 9.0 / 5.0 + 32.0; // conversion
  printf("Temperature in Fahrenheit: %d\n", degreeF);
}
~
-- INSERT --                               13,2-5         Bot
```

- `vim` program executes the commands in `.vimrc` every time it is invoked.
- Please copy `~ee2310/.vimrc` to your home directory

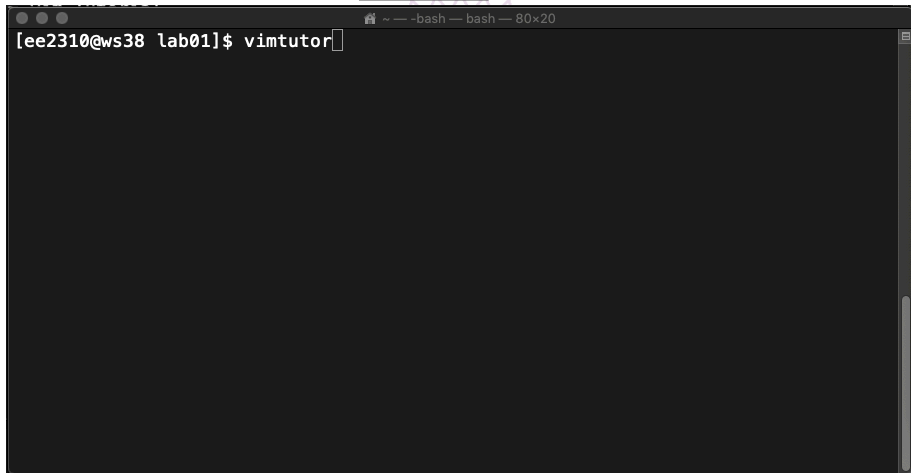


```
~/j2019/PL/labs — bash — bash — 63x10  
[ee2310@ws38 lab01]$ cp ~ee2310/.vimrc ~
```

- This file sets
  - Auto-indent mode
  - Each `Tab` inserts 4 spaces

# vim Tutorial

- `vim` program provides a tutorial for users to learn the easy commands
- At a linux terminal type in `vimtutor` as following to enter the tutorial



```
[ee2310@ws38 lab01]$ vimtutor
```

The screenshot shows a terminal window with a dark background. The prompt is `[ee2310@ws38 lab01]$` and the command `vimtutor` has been entered. The terminal title bar shows `~ - bash - bash - 80x20`. There are three window control buttons (red, yellow, green) in the top left corner of the terminal window.

- Most frequently used commands are demonstrated

```
~ - vim - vimtutor - bash - 80x20
=====
=  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!
```