



# Introduction to Vim, I

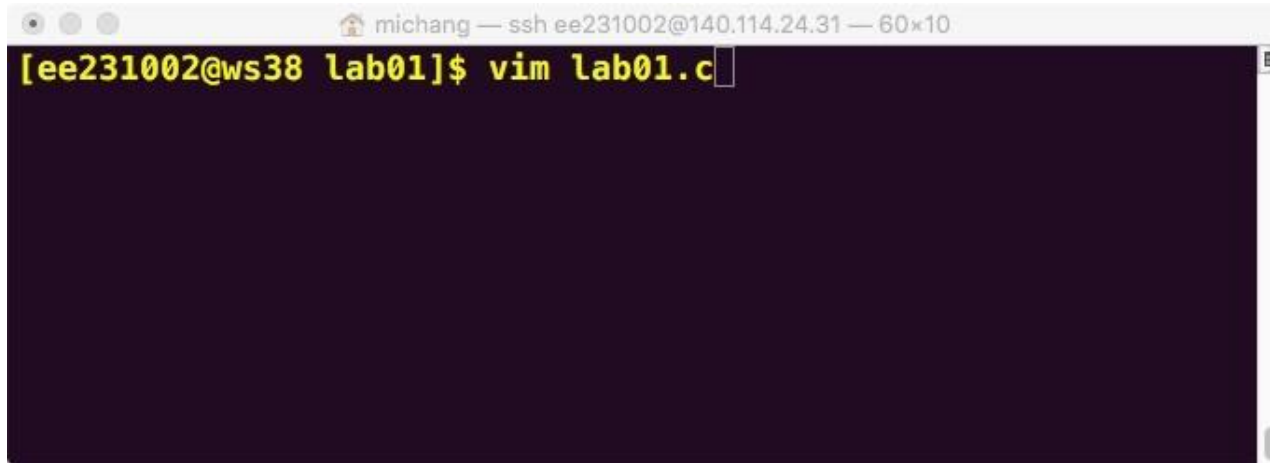
EE231002 Introduction to Programming

TA 胡恩典

Sep. 17, 2018

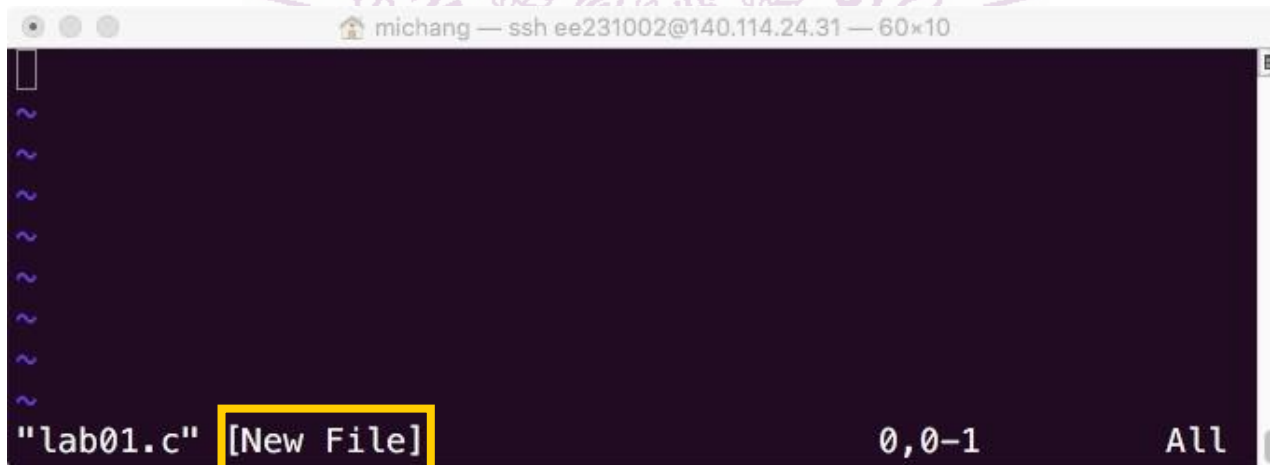
# Starting vim

- To start vim: `vim file`



A terminal window titled "michang — ssh ee231002@140.114.24.31 — 60x10". The prompt is "[ee231002@ws38 lab01]\$". The command "vim lab01.c" has been entered and is highlighted with a yellow box.

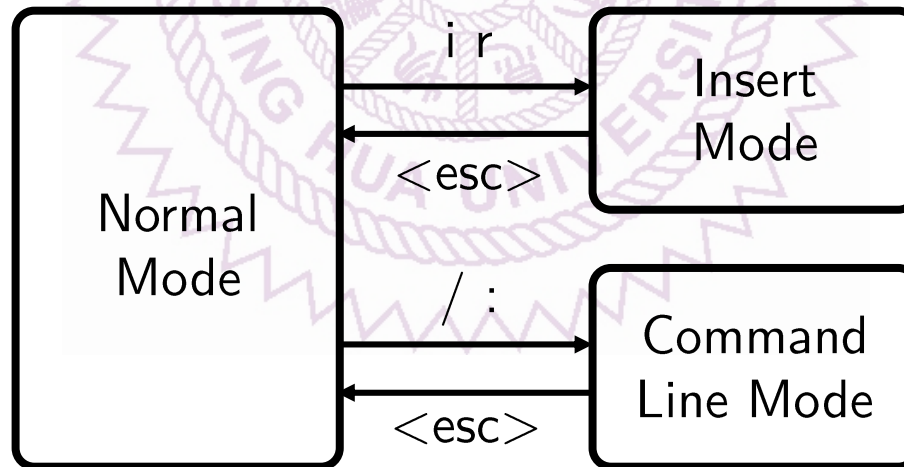
- For a new file



A terminal window titled "michang — ssh ee231002@140.114.24.31 — 60x10". The vim editor interface is shown. The file name "lab01.c" is visible in the bottom left, and "[New File]" is highlighted with a yellow box. The bottom right shows "0,0-1" and "All".

# 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
  - Normal mode and Command line mode will be covered next time








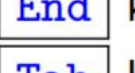






# Inserting Text

- 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




# Insert Mode

- In insert mode, you can type in texts
- To move cursor
  - , , ,  keys move cursor in four directions
  -  and  keys scroll one page of text
  -  key moves cursor to the beginning of the line
  -  key moves cursor to the end of the line
  -  key moves cursor to fixed columns (4x or 8x)
    - In our labs please use  key for indentation and each  key moves 4 spaces
- Press  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



The screenshot shows a terminal window with a vim editor. The editor contains a C program that prompts for a temperature in Celsius, converts it to Fahrenheit, and prints the result. The command `:wq` is entered in the command line at the bottom left, highlighted with a red circle. The terminal title bar shows the user 'michang' connected via SSH to 'ee231002@140.114.24.112' with a window size of '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

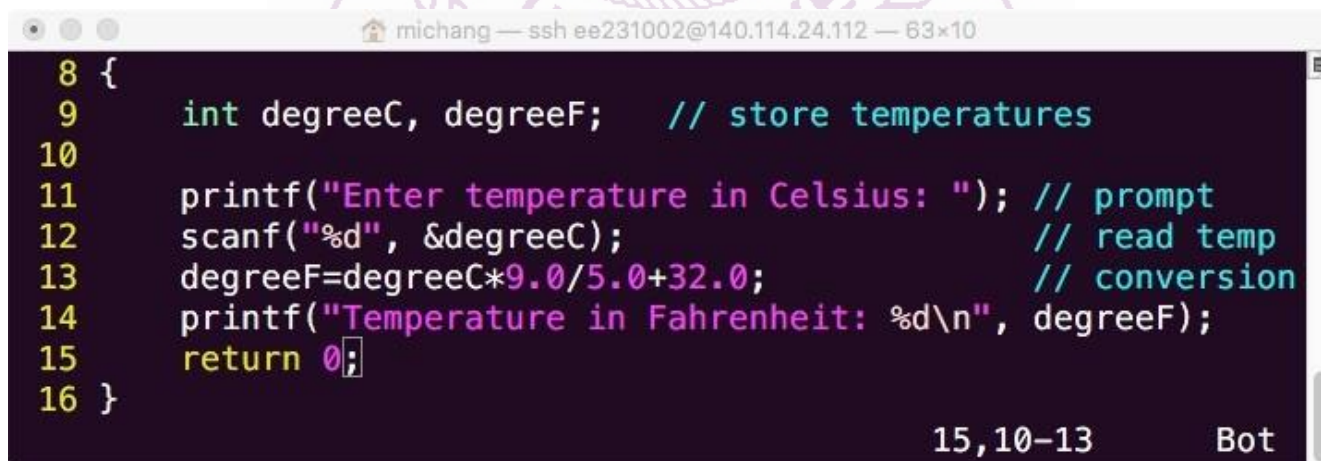


A screenshot of a vim editor window. The title bar shows 'michang — ssh ee231002@140.114.24.112 — 63x10'. The editor contains a C program for temperature conversion. The code is as follows:

```
{
    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;
}
```

The cursor is at the end of the line `:set nu` in normal mode.

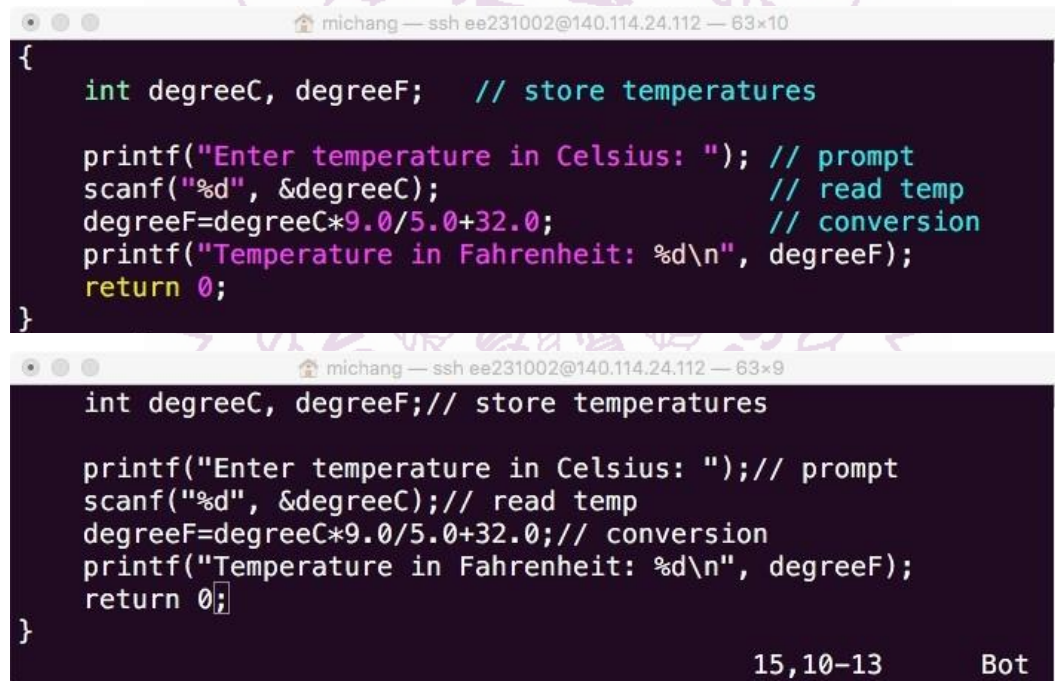


A screenshot of a vim editor window, identical to the one above but with line numbers enabled. The code is now displayed with line numbers from 8 to 16. At the bottom right of the editor, the status line shows '15,10-13' and 'Bot'.

```
8 {
9     int degreeC, degreeF;    // store temperatures
10
11     printf("Enter temperature in Celsius: "); // prompt
12     scanf("%d", &degreeC);                // read temp
13     degreeF=degreeC*9.0/5.0+32.0;          // conversion
14     printf("Temperature in Fahrenheit: %d\n", degreeF);
15     return 0;
16 }
```

# 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 the color text
- It also works in other kind of files like .cpp, .py, .js ...



The image shows two terminal windows side-by-side, both displaying the same C code. The top window has a title bar that reads 'michang — ssh ee231002@140.114.24.112 — 63x10'. The code in this window is color-coded: keywords like 'int', 'printf', 'scanf', 'return', and 'if' are in blue, comments are in green, and variable names like 'degreeC' and 'degreeF' are in yellow. The bottom window has a title bar that reads 'michang — ssh ee231002@140.114.24.112 — 63x9'. The code in this window is the same as the top one, but it is not color-coded, appearing in plain white text on a black background. At the bottom right of the bottom window, the text '15,10-13 Bot' is visible.

```
{
    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;
}
```

```
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;
}
```

15,10-13 Bot

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



# Color Text, II

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

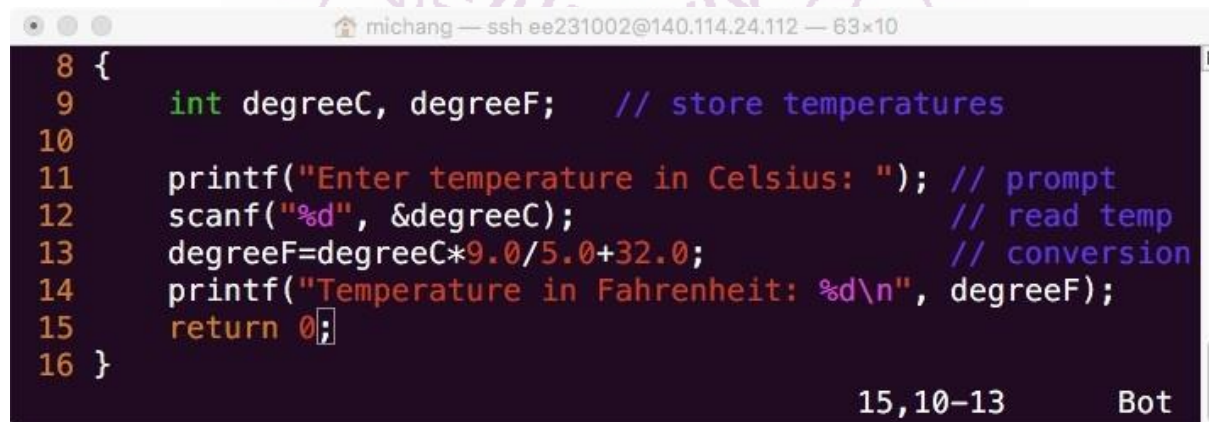
- `:set bg=dark`



```
8 {
9     int degreeC, degreeF;    // store temperatures
10
11     printf("Enter temperature in Celsius: "); // prompt
12     scanf("%d", &degreeC);           // read temp
13     degreeF=degreeC*9.0/5.0+32.0;      // conversion
14     printf("Temperature in Fahrenheit: %d\n", degreeF);
15     return 0;
16 }
```

15,10-13 Bot

- `:set bg=light`

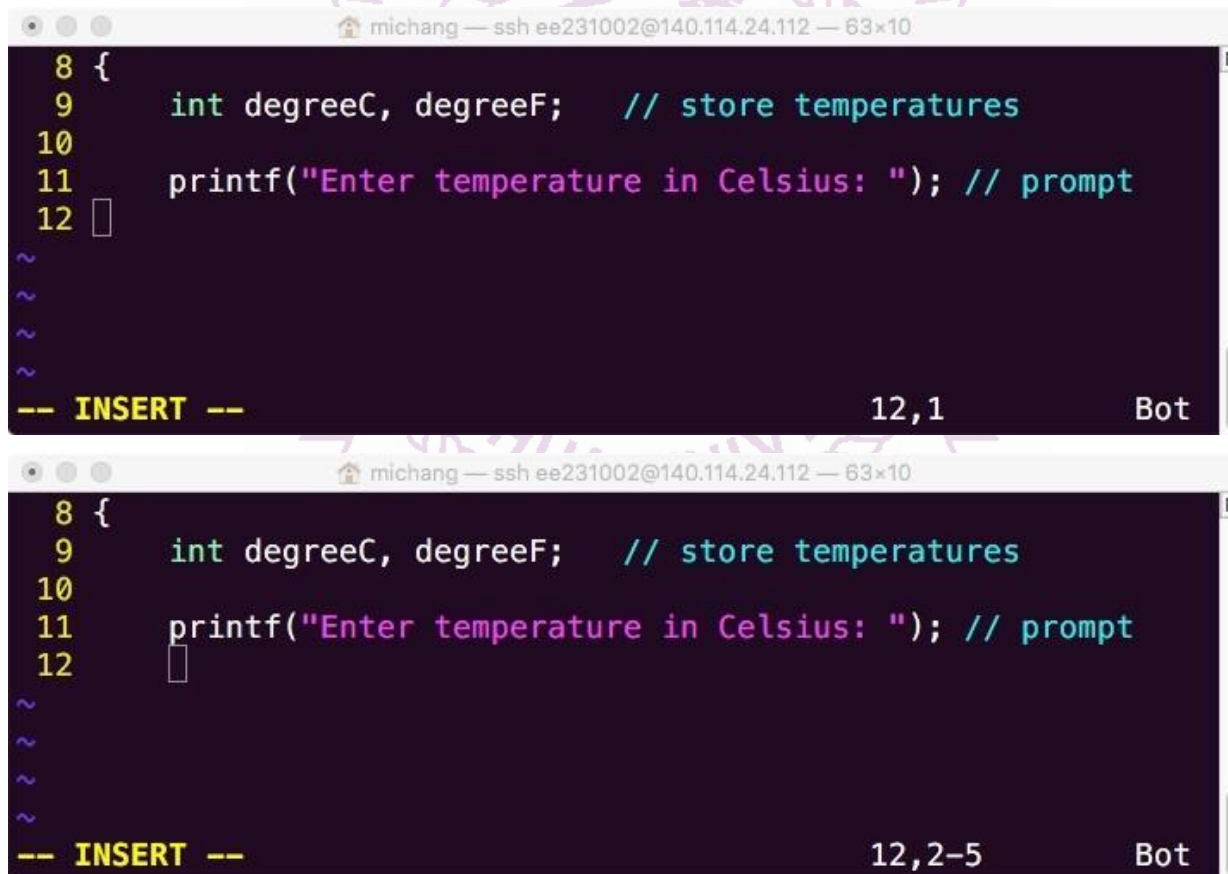


```
8 {
9     int degreeC, degreeF;    // store temperatures
10
11     printf("Enter temperature in Celsius: "); // prompt
12     scanf("%d", &degreeC);           // read temp
13     degreeF=degreeC*9.0/5.0+32.0;      // conversion
14     printf("Temperature in Fahrenheit: %d\n", degreeF);
15     return 0;
16 }
```

15,10-13 Bot

# 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

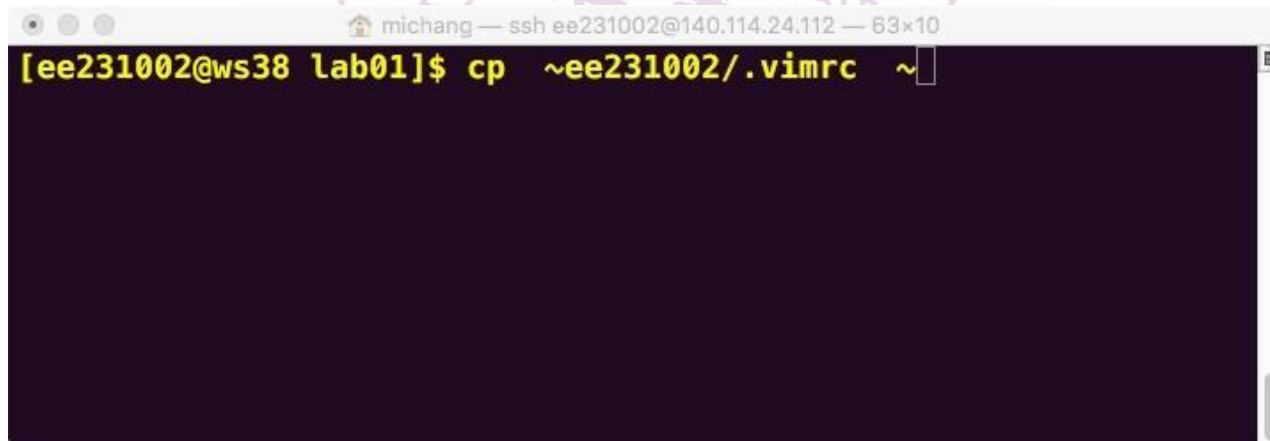


```
michang — ssh ee231002@140.114.24.112 — 63x10
8 {
9     int degreeC, degreeF;    // store temperatures
10
11     printf("Enter temperature in Celsius: "); // prompt
12
-- INSERT --                               12,1                               Bot
```

```
michang — ssh ee231002@140.114.24.112 — 63x10
8 {
9     int degreeC, degreeF;    // store temperatures
10
11     printf("Enter temperature in Celsius: "); // prompt
12
-- INSERT --                               12,2-5                               Bot
```

# .vimrc

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



A terminal window titled "michang — ssh ee231002@140.114.24.112 — 63x10". The prompt is "[ee231002@ws38 lab01]\$". The command "cp ~ee231002/.vimrc ~" has been entered, and the cursor is at the end of the line.

- This file sets
  - Auto-indent mode
  - Each `Tab` inserts 4 spaces
  - Remember that your points will be deducted if you don't indent appropriately

# Set your preference in .vimrc

- Following example shows how to set your preference
- Before

```
set tabstop=4  
set sw=4
```

.vimrc

```
#include <stdio.h>  
  
int main(void)  
{  
    return 0;  
}
```

lab01.c

- Launch `.vimrc` by command `vim ~/.vimrc`
- Insert `set nu` and you will see the difference after saving by `:wq`

```
set tabstop=4  
set sw=4  
set nu
```

.vimrc

```
1 #include <stdio.h>  
2  
3 int main(void)  
4 {  
5     return 0;  
6 }
```

lab01.c

# vim Tutorial

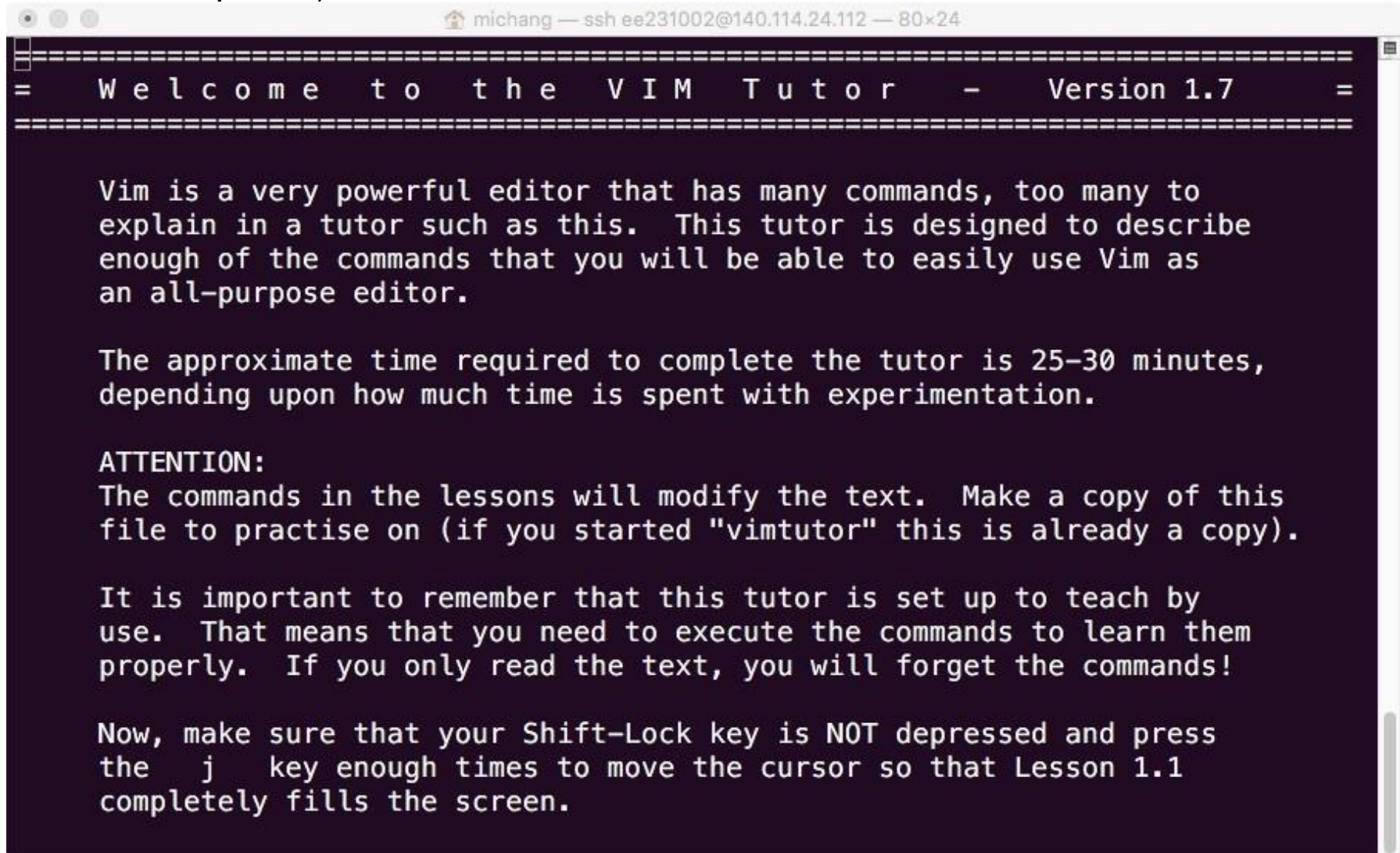
- `vim` program provides a tutorial for users to learn the easy commands
- At a terminal type in `vimtutor` as following to enter the tutorial



A screenshot of a terminal window. The title bar at the top shows a home icon, the username 'michang', and the connection details 'ssh ee231002@140.114.24.112' followed by the window size '80x20'. The terminal itself has a dark purple background. The prompt is '[ee231002@ws38 lab01]\$' in yellow text. The command 'vimtutor' is being typed in white text, with a white cursor at the end of the word.

# vim Tutorial, II

- Most frequently used commands are demonstrated



```
michang — ssh ee231002@140.114.24.112 — 80x24
=====
=  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 practise 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.
```



# Compiler and GCC

- GCC: GNU Compiler Collection
  - Original: GNU C Compiler for processing C only
  - It can process some other programming languages like C++ nowadays
- Compiler: transforms code in one programming language to target language
  - Just consider it as a translator
- C Compiler performs following operation when compiling
  - Lexical Analysis: Mark the tokens like keywords, constants, identifiers...
  - Parsing: Go through the tokens and process the grammar
  - Code Generation: For C Compiler, it generates the assembly code
  - Linking: Link the related object file and generate an execution file
  - Load: Load the execution file into the memory
- In this course, you only need to know:
  - `gcc labxx.c` generates an execution file `a.out`
  - `./a.out` executes your program
- For more information about Compiling and Compiler, you can take:  
CS1356 I2P(II), EE3450/CS4100 Computer Archi. CS3404 Compiler Design

# Example Program in Required Format

Format.c

```
1  /* EE231002 Lab00 Format
2   * 104061220 En-Tien, Hu
3   * Date: Sep. 17, 2018
4   */
5
6  #include <stdio.h>
7
8  int main(void)
9  {
10     int i;                //for iteration
11     int id;               //to store the input
12
13     printf("Insert your student ID: "); //print the statement
14     scanf("%d", &id);      //read the input number
15
16     for (i = 0; i < 10; i++) { //10-time iteration
17         if (i % 2 == 0)        //if i is multiple of 2
18             printf("%d\n", id); //print the id
19         else                   //if not
20             printf("%d\n", id + 100); //print id+100
21     }
22
23     return 0;              //program terminates
24 }
```

# Appendix

EE231002 Introduction to Programming

TA 胡恩典 (logical203010@gmail.com)

(Optional) Send an email to me for following information

# For those who want more practice

- Google “NTHU Online Judge”
- This is used in CS Department, but everyone can register an account
- Click “Contest”, you can try some of which beginning with I2P or I2P(I)
- Actually, you don’t need any extra practice in our course
  - Therefore, don’t spend too much time on this
  - All you need to do is getting familiar with problems in our lab



## NTHUOJ

This is National Tsing Hua University Online Judge, an ACM-like online judge designed for training purposes and a platform of contest. You may utilize this site by the links on the side bar and top-right corner. Each of them will redirect you to a page with the following functions.

### Announcements

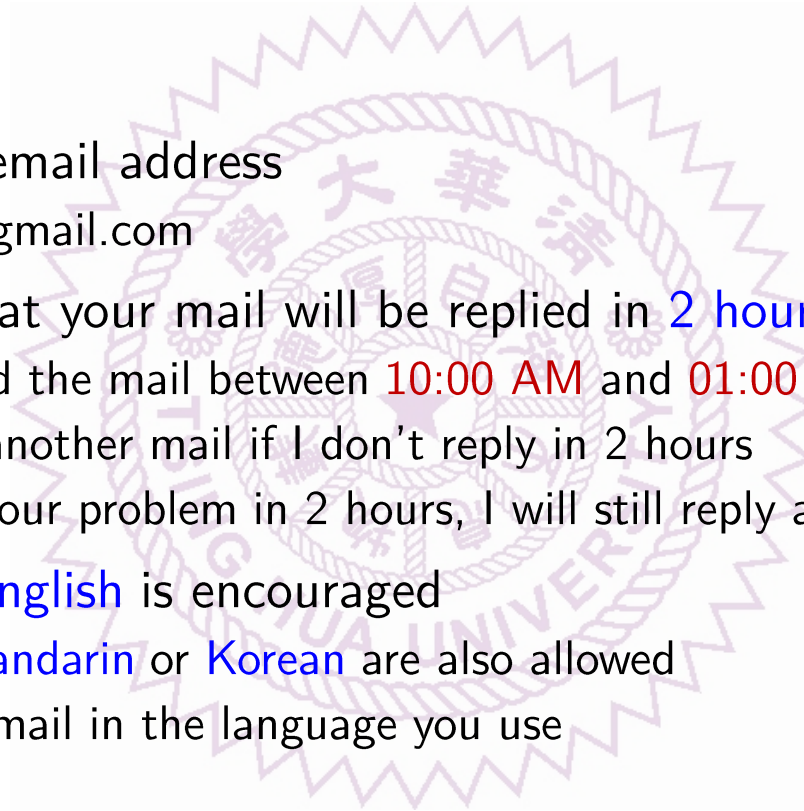
Any bug? Report here: <http://goo.gl/forms/jOeiwZdjku>  
If you have any problem with your account, please email to [nthucsoj@gmail.com](mailto:nthucsoj@gmail.com)  
JAVA and Python update: <https://goo.gl/2tord8> (2018/04/18)

### Running / Upcoming Contest

Register	Contest name	Remaining time	Upcoming
----------	--------------	----------------	----------

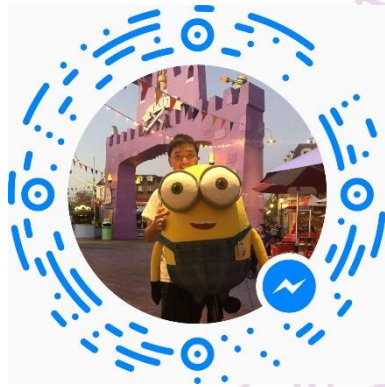
# My Email Policy

- Again, this is my email address
  - logical203010@gmail.com
- You can expect that your mail will be replied in 2 hours
  - Only if you send the mail between 10:00 AM and 01:00 AM
  - You may send another mail if I don't reply in 2 hours
  - If I can't solve your problem in 2 hours, I will still reply and ask you to wait
- Writing mails in English is encouraged
  - But mails in Mandarin or Korean are also allowed
  - I will reply the mail in the language you use



# Links to SNS Accounts and Policy

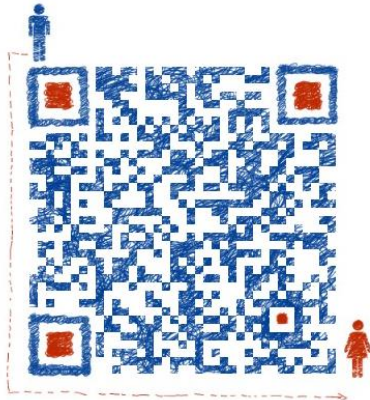
- Following SNS Accounts are provided for convenience
  - We don't encourage you to ask question through SNS
  - Practicing formal letters is important



Messenger



Kakao Talk



WeChat

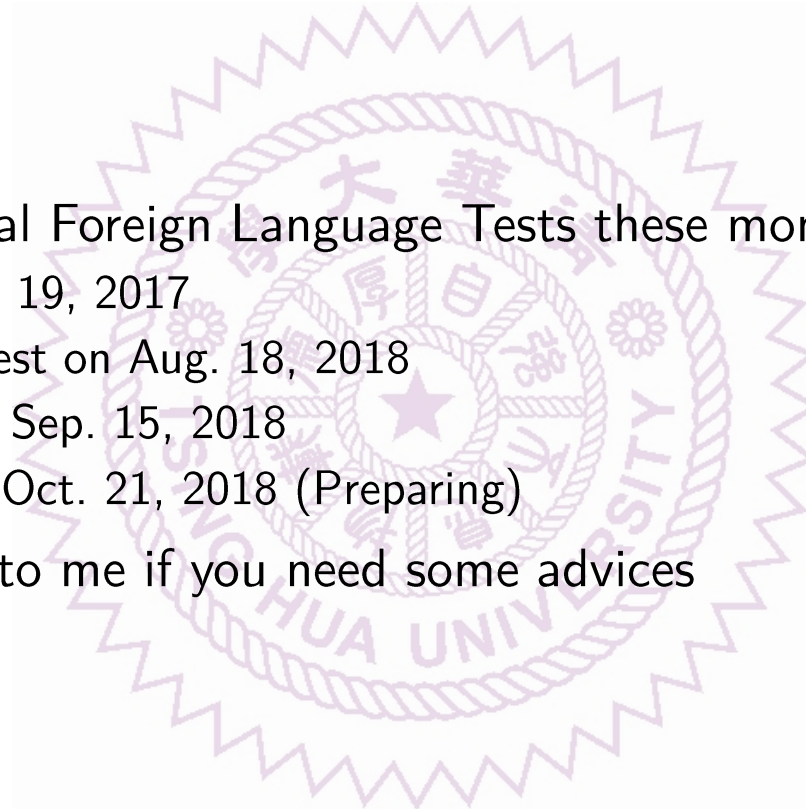


LINE



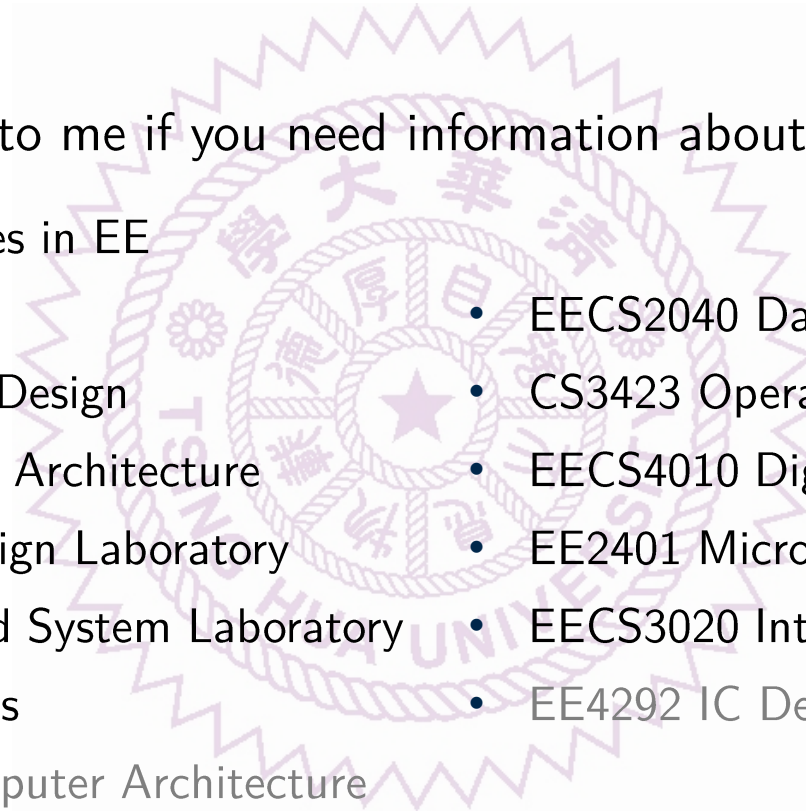
# About Foreign Language Test

- I have taken several Foreign Language Tests these months
  - TOEIC on Dec. 19, 2017
  - GRE General Test on Aug. 18, 2018
  - TOEFL iBT on Sep. 15, 2018
  - TOPIK I, II on Oct. 21, 2018 (Preparing)
- Feel free to come to me if you need some advices

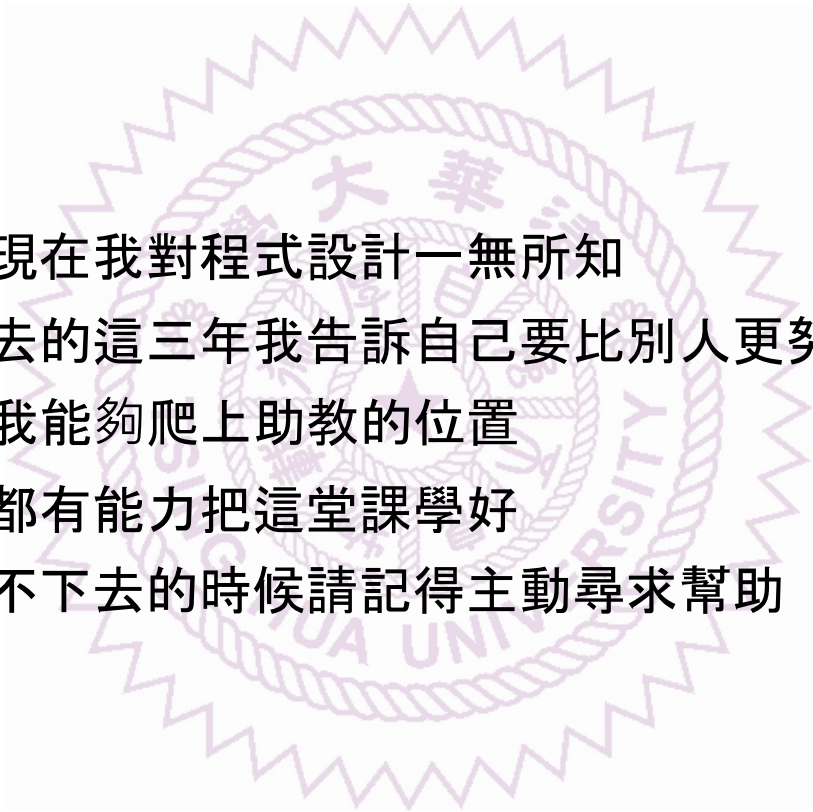


# Courses I have taken in EECS

- Feel free to come to me if you need information about following courses
  - All required courses in EE
  - CS1356 I2P(II)
  - CS3404 Compiler Design
  - CS4100 Computer Architecture
  - EE2230 Logic Design Laboratory
  - EE2405 Embedded System Laboratory
  - EE3980 Algorithms
  - EE6455 Adv. Computer Architecture
  - EECS2040 Data Structure
  - CS3423 Operating Systems
  - EECS4010 Digital System Design
  - EE2401 Microprocessor Systems
  - EECS3020 Intro. to Computer Network
  - EE4292 IC Design Laboratory



# Encouragement from TA



三年前的現在我對程式設計一無所知  
所以在過去的這三年我告訴自己要比別人更努力  
既然今天我能夠爬上助教的位置  
表示大家都有能力把這堂課學好  
當你快撐不下去的時候請記得主動尋求幫助

—TA