LINUX AND C INTRODUCTION

曾歆勋

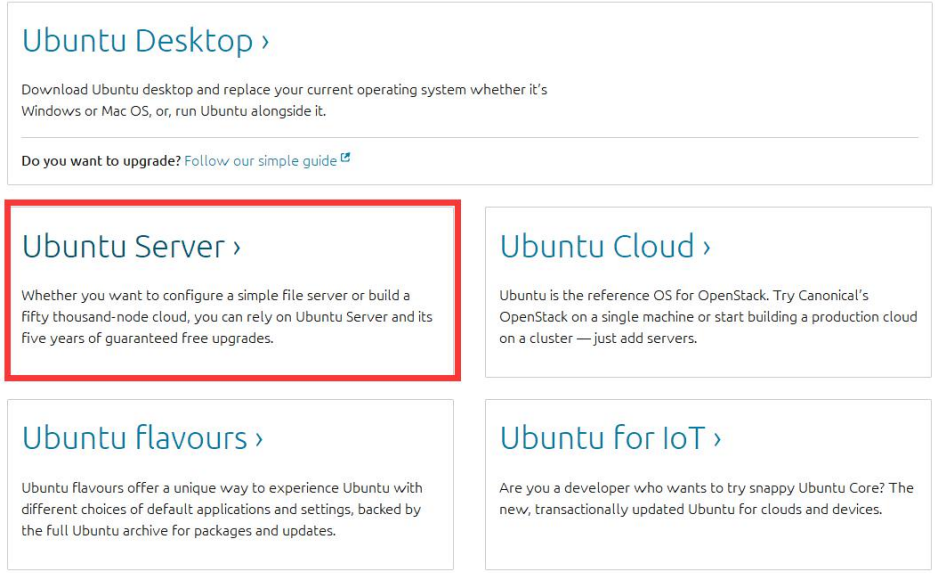
鸣谢黄博

GETTING STARTED

Pictures from google

HOW TO GET A LINUX OS?

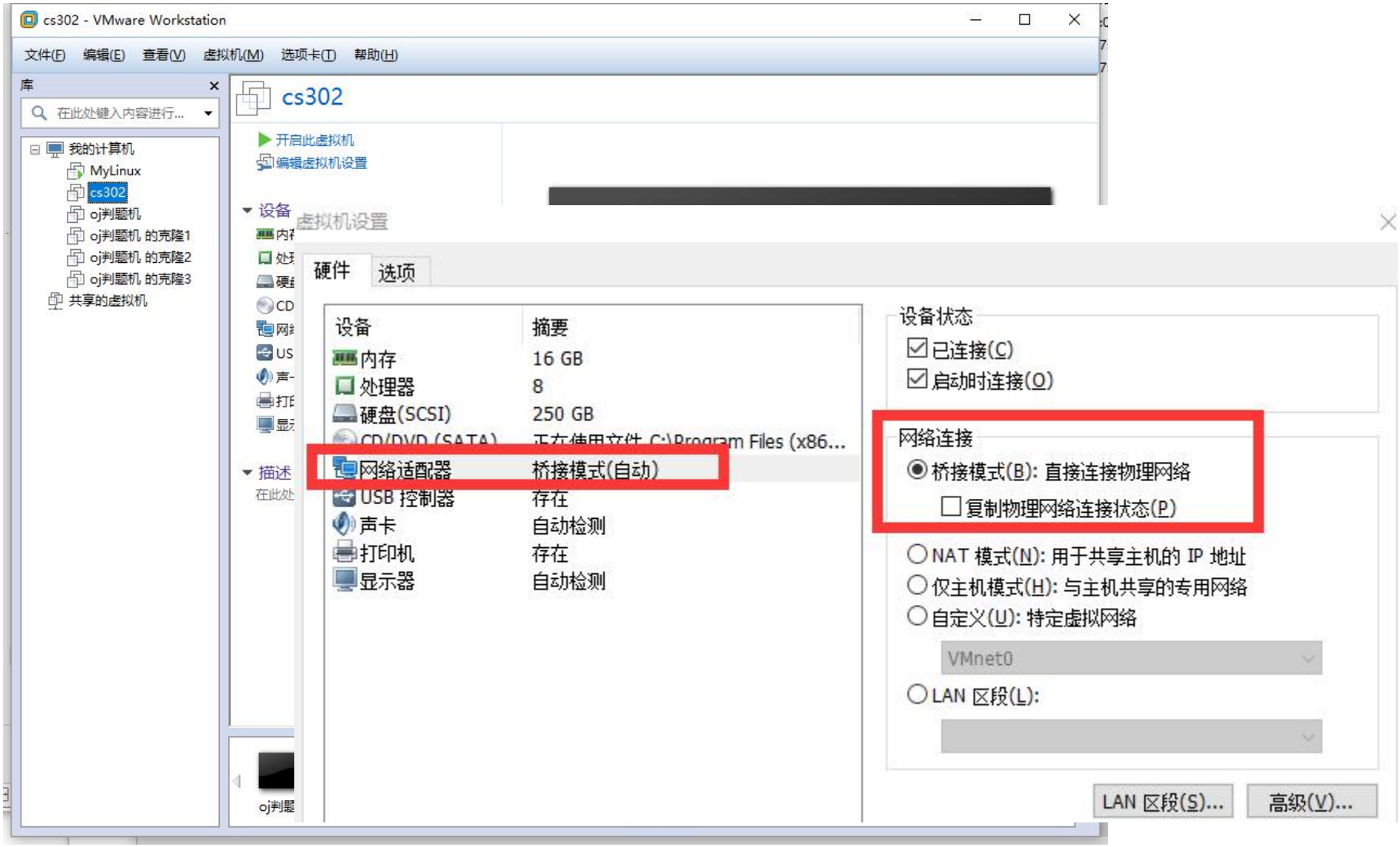
- **Ubuntu Server 18.04 LTS recommend**
  - https://www.ubuntu.com/download
  - Burn the iso into your USB. (ultraiso)
  - Startup by USB
  - Install



HOW TO GET A LINUX OS?

- **Using virtual machine**
  - Download Vmware

VIRTUAL MACHINE



BASIC COMMAND

Pictures from google

BASIC COMMAND

- **man xxx**
  - show the manual of command xxx
  - you can try “man man”

BASIC COMMAND

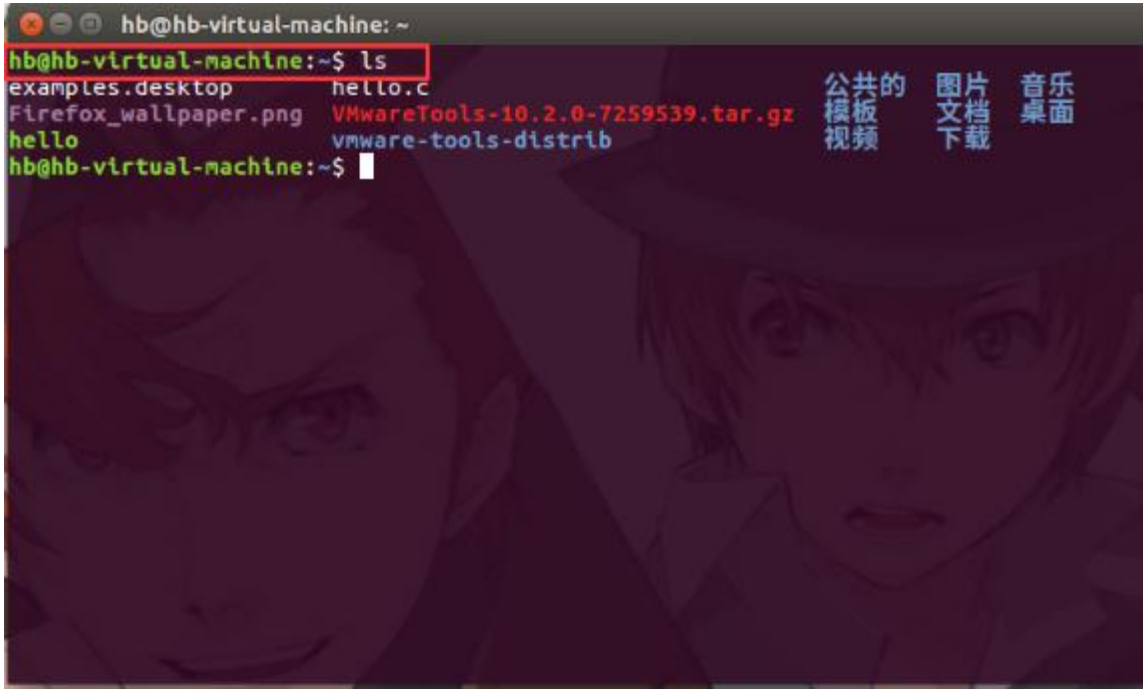
- **ls**
  - list directory contents
  - let's try "man ls"

BASIC COMMAND

- **mkdir**
  - make a new directory
  - try man mkdir by yourself
  - try to make a directory named OS in HOME(~)

BASIC COMMAND

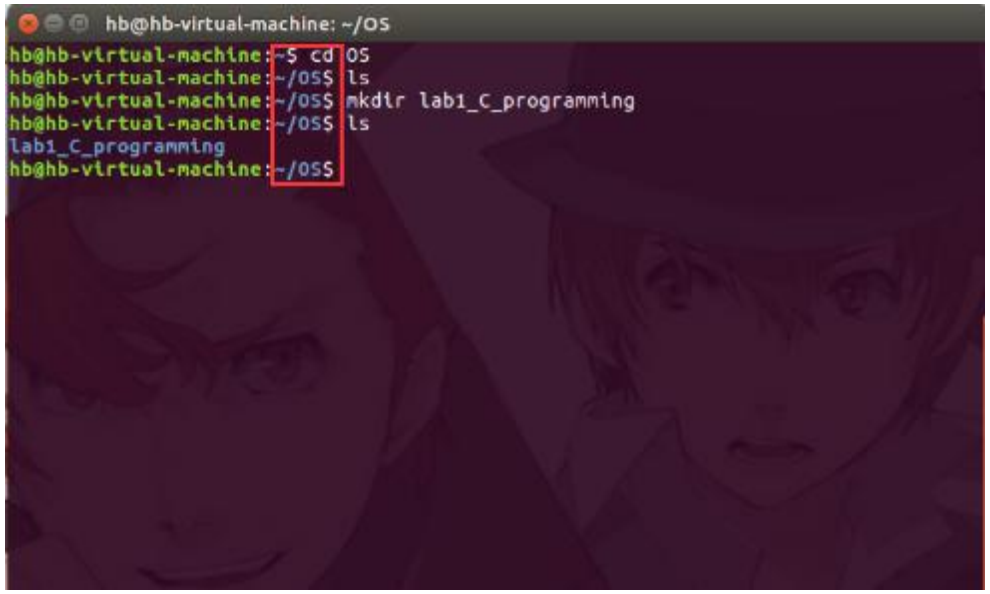
- **ls**



- How to go into OS?

BASIC COMMAND

- **cd**
  - change directory
  - let’s go to OS directory
  - Try cd /home
  - Try cd ~, cd ..



BASIC COMMAND

- **apt-get install vim**
  - This command need root authority. Use sudo to swtich get root authority for a while.
  - apt-get handling packages
  - install means we want to install this package
  - you can man apt-get to learn details

EDITOR

Pictures from google

WHY WE NEED EDITOR

- **Server**
  - When you connect to a linux server, sometimes it doesn’t have GUI.

VIM

- **vim**
  - A powerful editor.
  - You can use vim/vi in terminal to edit files.
  - In order to get full functions about vim, we can install some packages first.

CONFIGURE VIM

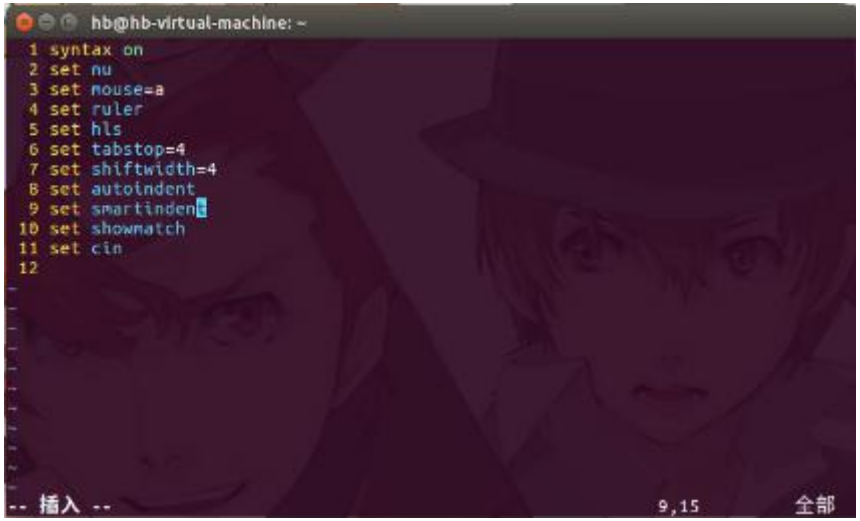
- This is not necessary, just let you be more comfortable when using vim.
- Go to HOME
- Using vim command to edit file .vimrc

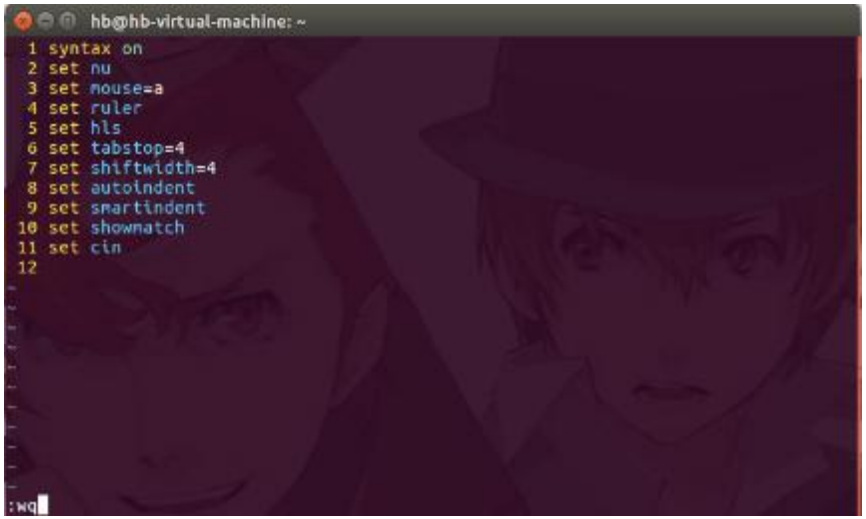
VIM

- **Vim has three modes, they are:**
  - Command mode: you can not input text, everything you input will be command.
  - Insert mode: you can input text. Press Esc to return command mode.
  - Last line mode: you can input special command. Such as exit and find string.

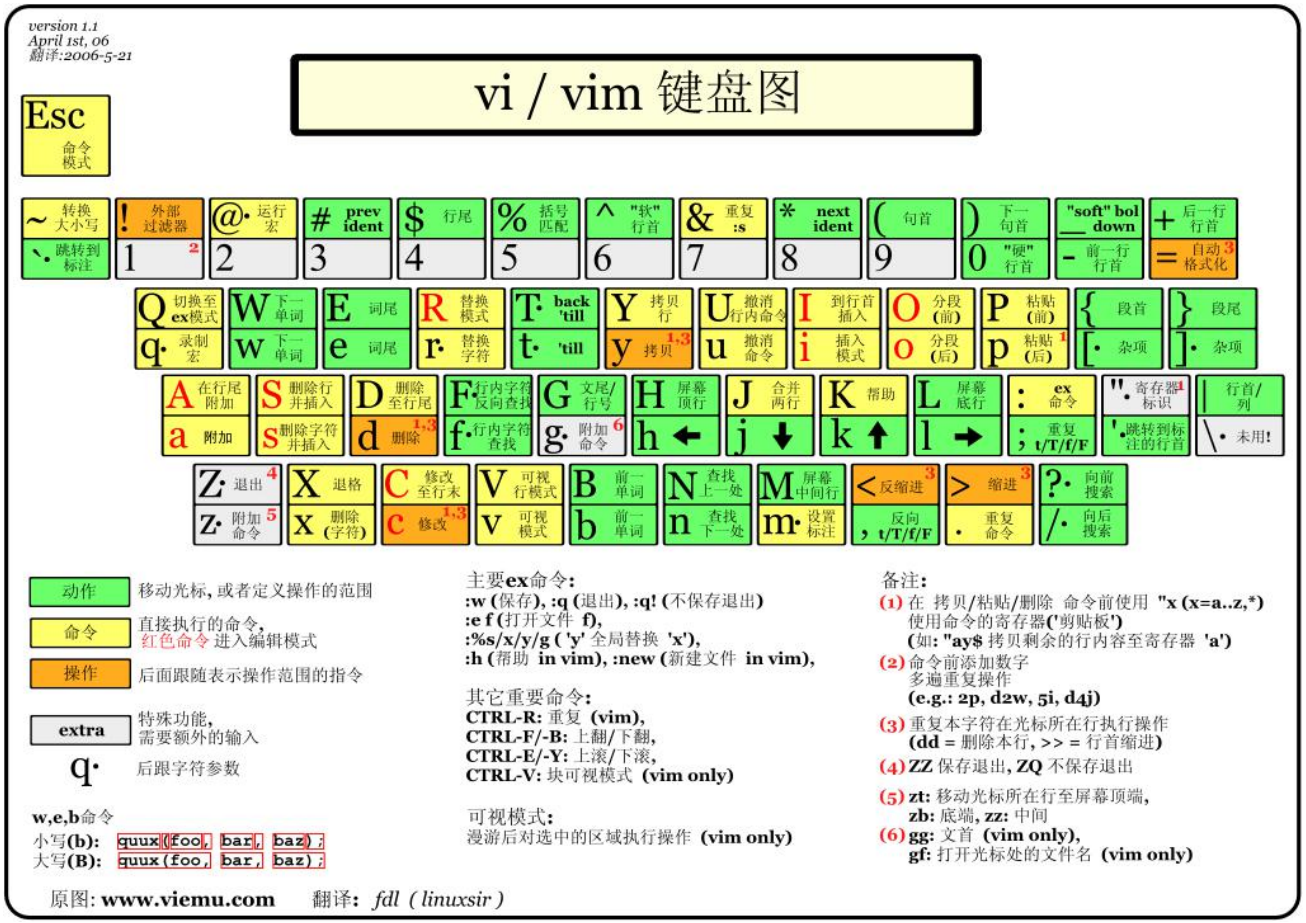
VIM

- **Configure**
  - Press i to go to insert mode
  - Input text
  - Press Esc go back to command mode
  - Press shift + ; to go to last line mode
  - Press wq to write and quit

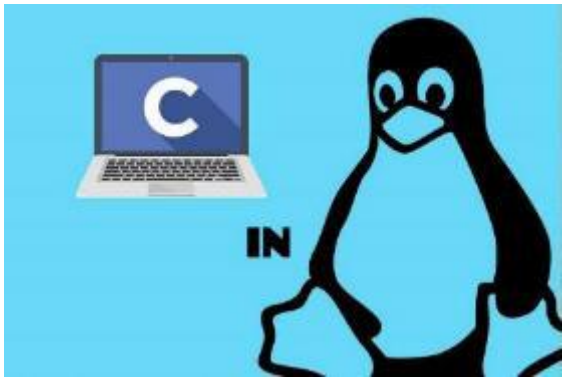




YOU MAY NEED THIS



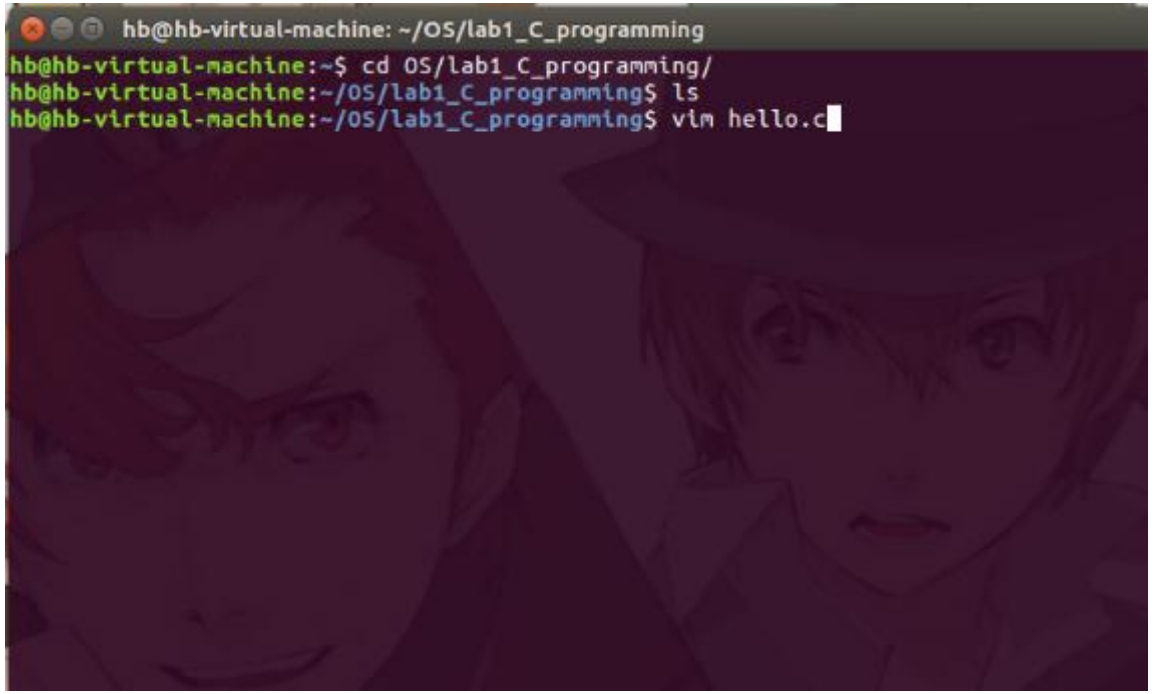
PROGRAMMING



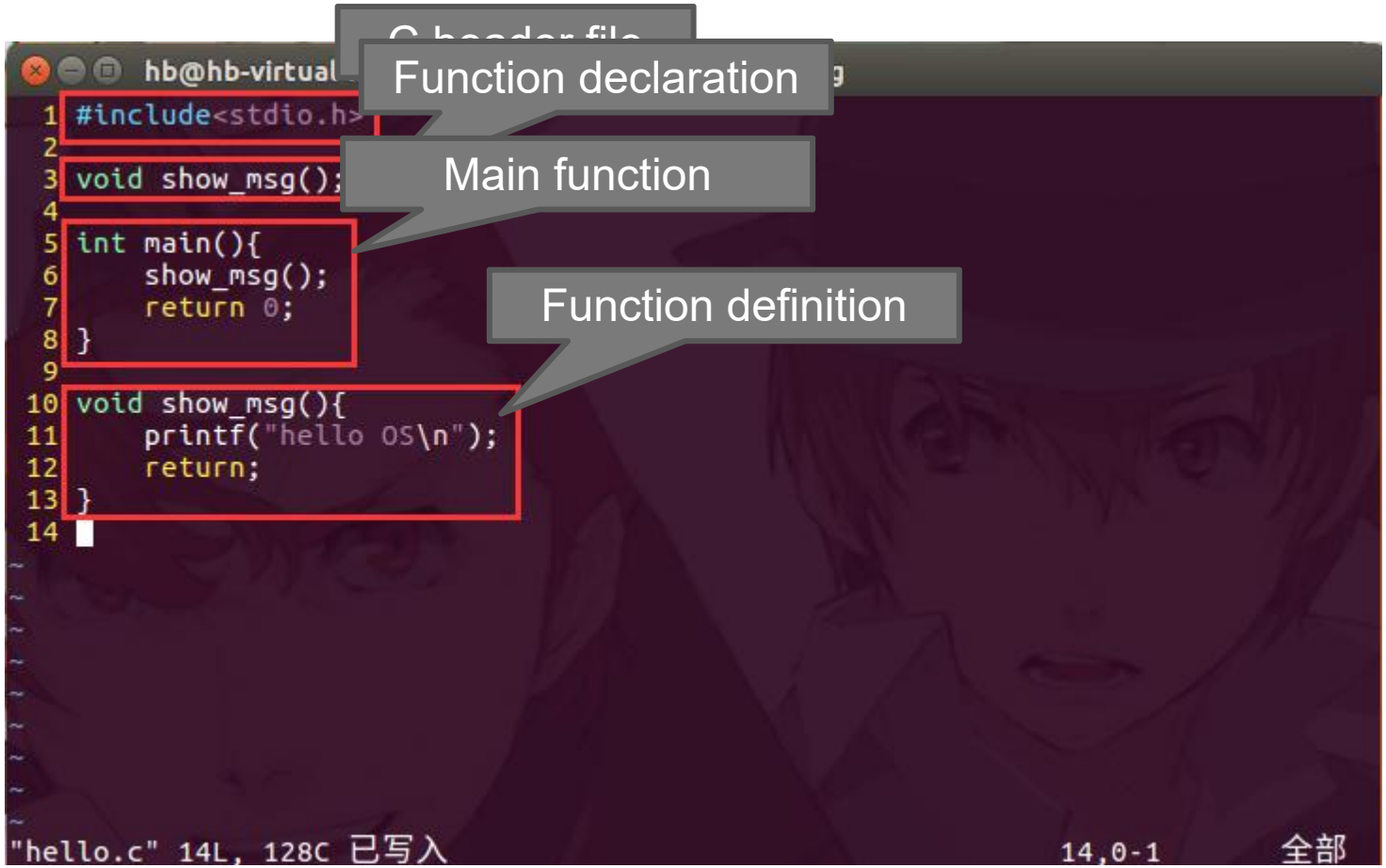
Pictures from google

FIRST C PROGRAM

- **Now we are ready for our first C program.**
  - Go to lab1_C_programming directory
  - And edit file: hello.c



FIRST C PROGRAM



A JAVA PROGRAM

hb@hb-virtual-machine: ~/OS/lab1_C_programming

```java
import java.io.*;
import java.util.*;

public class HelloOS{

    public static void print_msg(){
        System.out.println("hello OS");
    }

    public static void main(string[] args){
        print_msg();
    }
}

"hello.java" 14L, 195C 已写入
```

hb@hb-virtual-machine: ~/OS/lab1_C_programming

```c
#include<stdio.h>

void show_msg();

int main(){
    show_msg();
    return 0;
}

void show_msg(){
    printf("hello OS\n");
    return;
}

"hello.c" 14L, 128C                    14,0-1        全部
```

FIRST C PROGRAM

- **How to run our program?**
  - We need compile it!
  - Ubuntu has GCC (GNU Compiler Collection)
  - Let’s compile our first c program

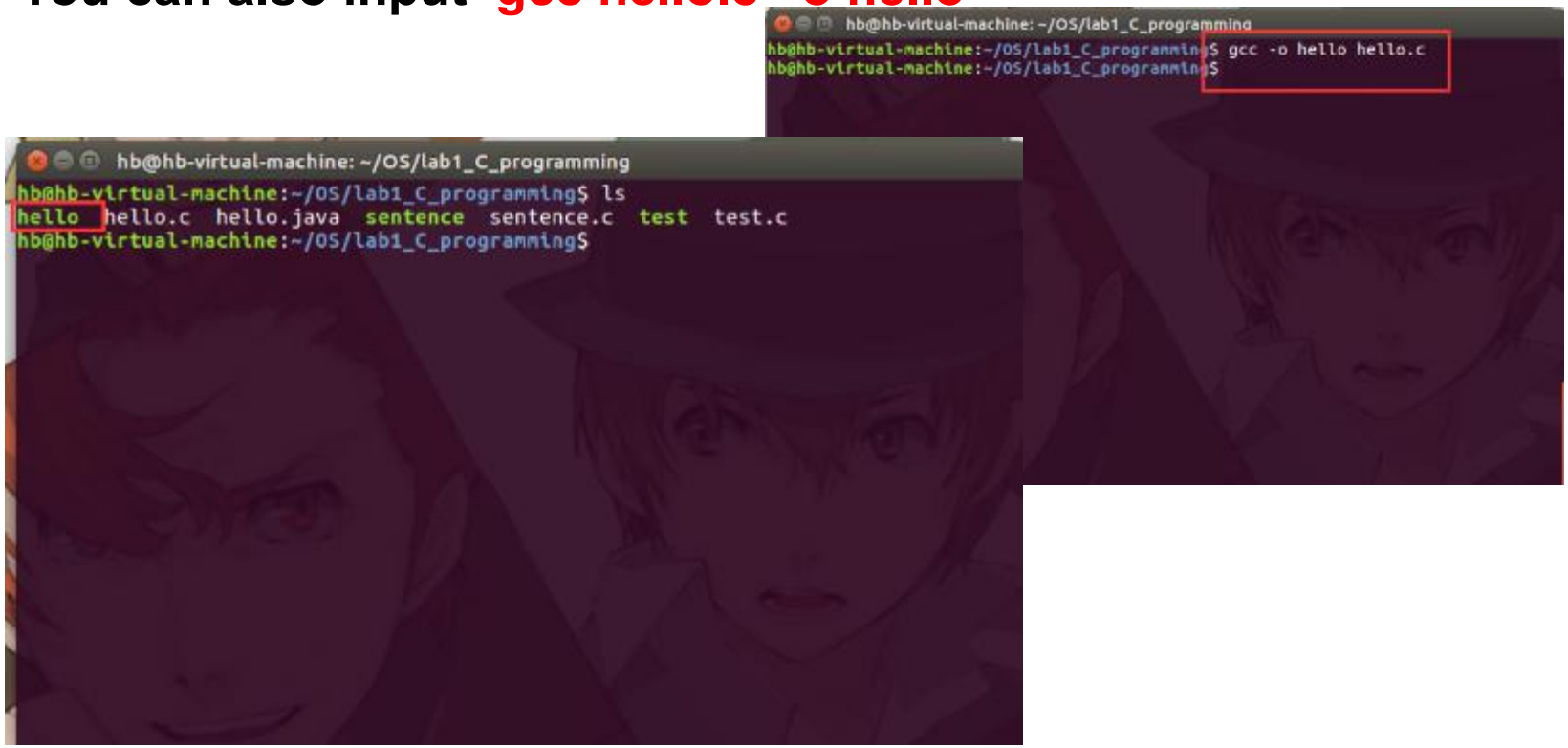
```
root@hb-virtual-machine: ~
GCC(1)                                  GNU                                  GCC(1)

NAME
       gcc - GNU project C and C++ compiler

SYNOPSIS
       gcc [-c|-S|-E] [-std=standard]
           [-g] [-pg] [-Olevel]
           [-Wwarn...] [-Wpedantic]
           [-Idir...] [-Ldir...]
           [-Dmacro[=defn]...] [-Umacro]
           [-foption...] [-mmachine-option...]
           [-o outfile] [@file] infile...

       Only the most useful options are listed here; see below for the
       remainder.  g++ accepts mostly the same options as gcc.

DESCRIPTION
       When you invoke GCC, it normally does preprocessing, compilation,
       assembly and linking.  The "overall options" allow you to stop this
       process at an intermediate stage.  For example, the -c option says not
       to run the linker.  Then the output consists of object files output by
       the assembler.
 Manual page gcc(1) line 1 (press h for help or q to quit)
```

ABOUT GCC

- **gcc**
  - -c Compile or assemble the source files, but do not link.
  - -S Stop after the stage of compilation proper; do not assemble.
  - -E Stop after the preprocessing stage; do not run the compiler proper.
  - -o filename Place output in file file.
  - If no parameters, gcc will do all things and output an execute file a.out

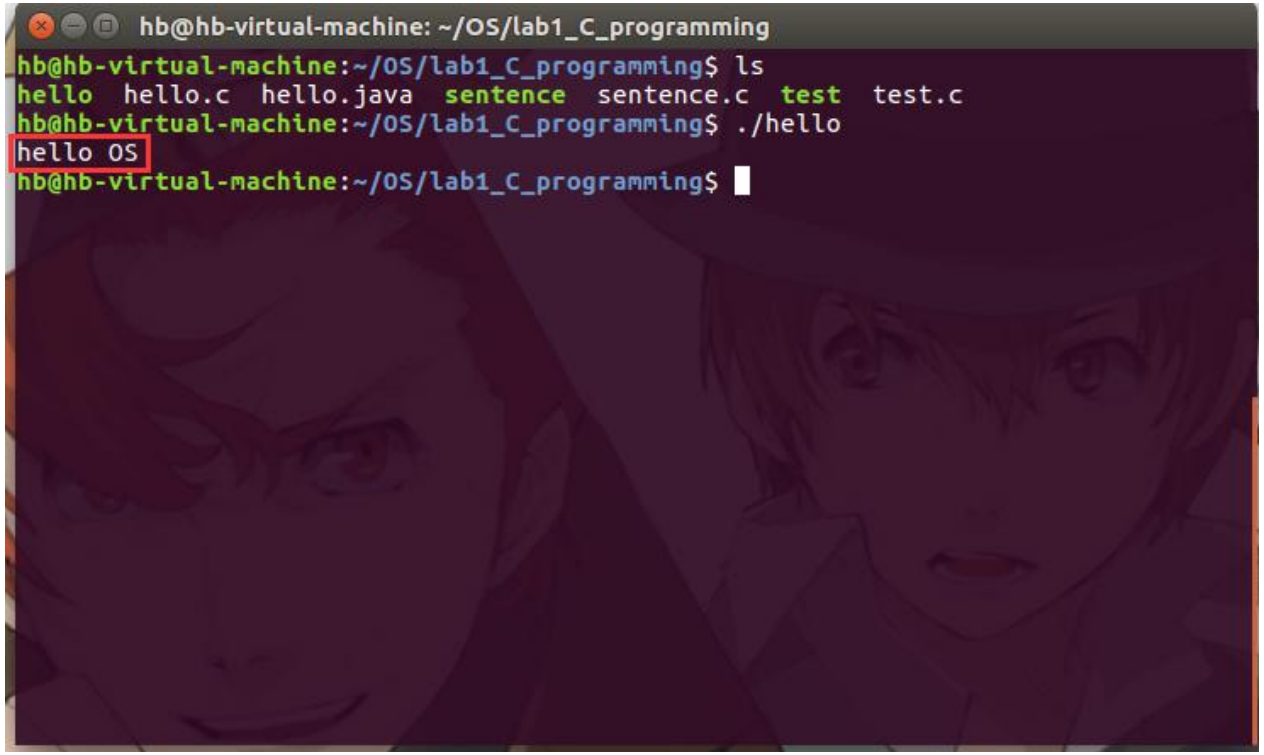
FIRST C PROGRAM

- Input gcc –o hello hello.c on terminal
- You can also input gcc hello.c –o hello

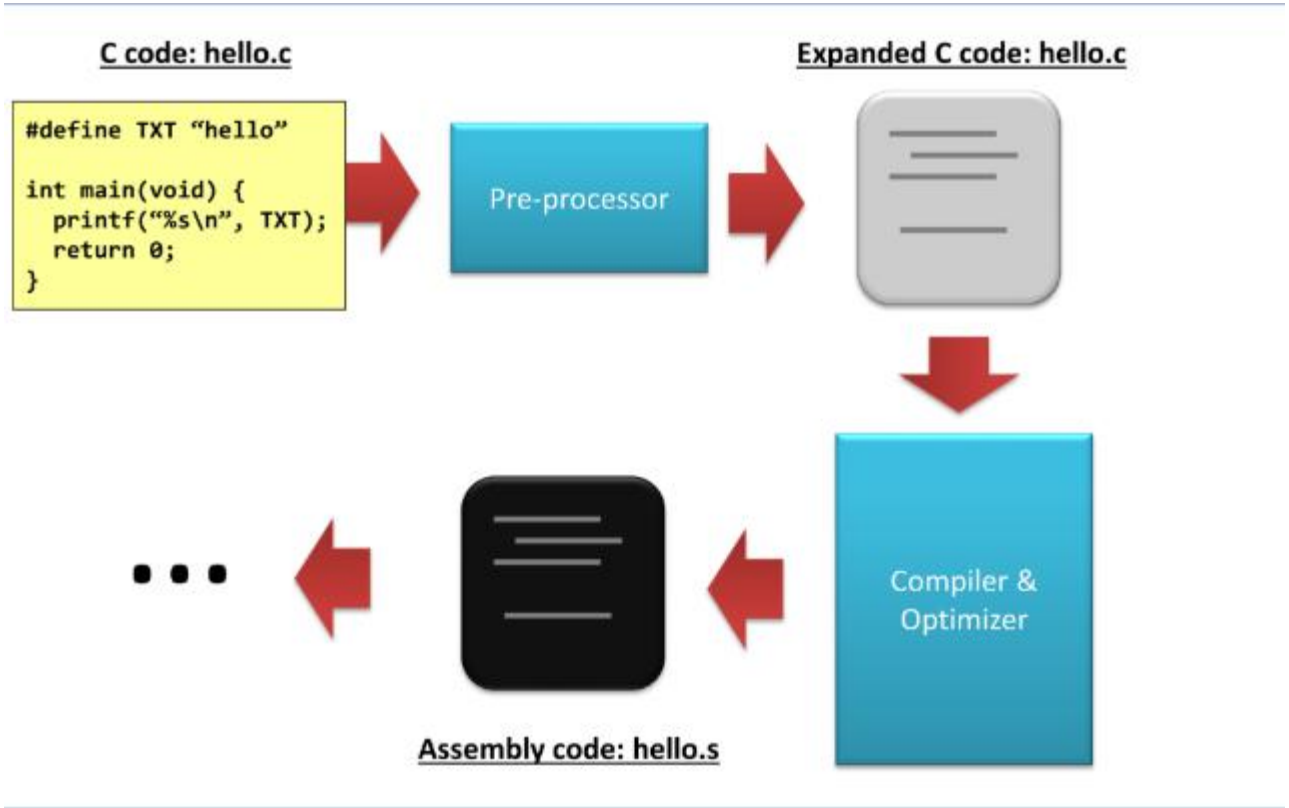


FIRST C PROGRAM

- **Let's run it!**
  - ./hello

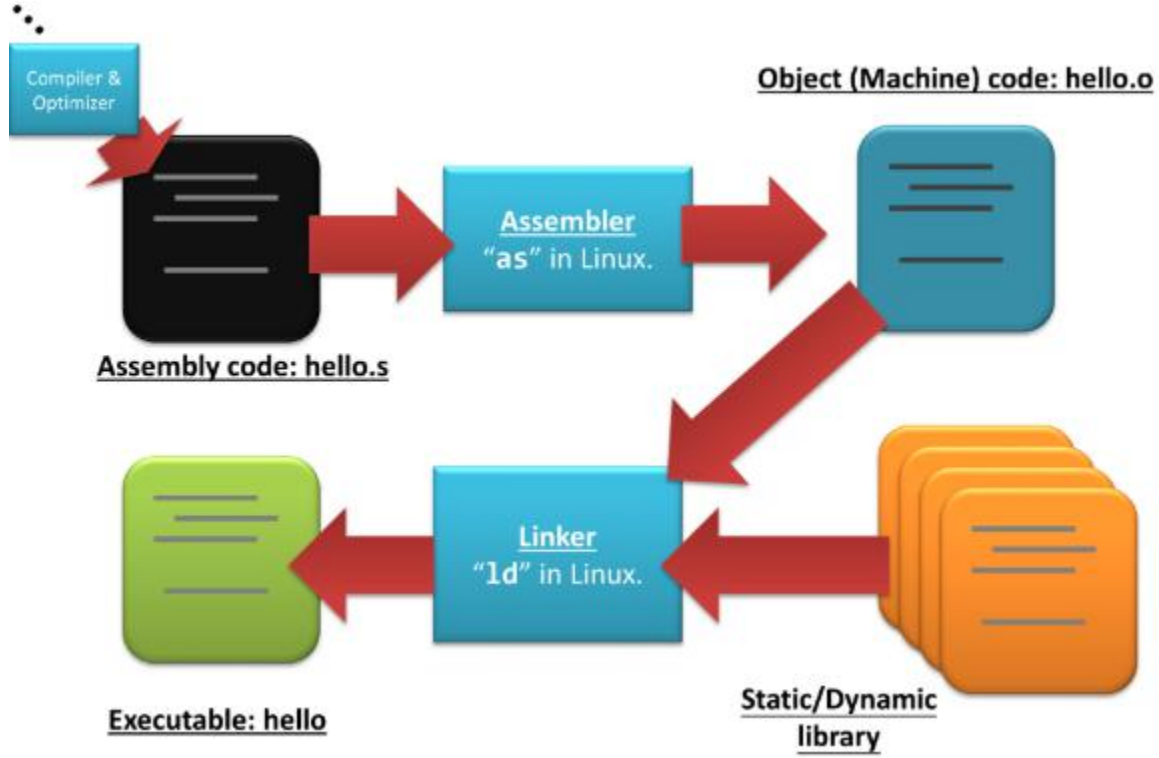


WHAT HAPPENS?



Pictures from: https://calvinkam.github.io/csci3150-Fall17-lab3/building-a-program.html

WHAT HAPPENS?



Pictures from: https://calvinkam.github.io/csci3150-Fall17-lab3/building-a-program.html

WHAT HAPPENS?

- **Pre-processor**
- **Input gcc –E hello.c**
  - Replace #include

```
extern int ftrylockfile (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__)
) ;


extern void funlockfile (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__)
);
# 942 "/usr/include/stdio.h" 3 4

# 2 "hello.c" 2


# 3 "hello.c"
void show_msg();

int main(){
 show_msg();
 return 0;
}

void show_msg(){
 printf("hello OS\n");
 return;
}
hb@hb-virtual-machine:~/OS/lab1_C_programming$
```

WHAT HAPPENS?

- **compiler and optimizer**
  - First check syntax and analyze it.
  - Then produce assembly code.
  - Optimizer will improve the code quality.

MORE ABOUT OPTIMIZER

- **Consider this example opt.c**

```
hb@hb-virtual-machine:~/OS/lab1_C_programming$ cat opt.c
#include<stdio.h>

int main(){
        int x = 0;
        x += 1;
        x += 1;
        x += 1;
        printf("%d\n", x);
        return 0;
}
```

- **We open the optimizer to get assembly code**

```
hb@hb-virtual-machine:~/OS/lab1_C_programming$ gcc -S opt.c -O0 -o opt0.s
hb@hb-virtual-machine:~/OS/lab1_C_programming$ gcc -S opt.c -O1 -o opt1.s
hb@hb-virtual-machine:~/OS/lab1_C_programming$
```

MORE ABOUT OPTIMIZER

opt0.s

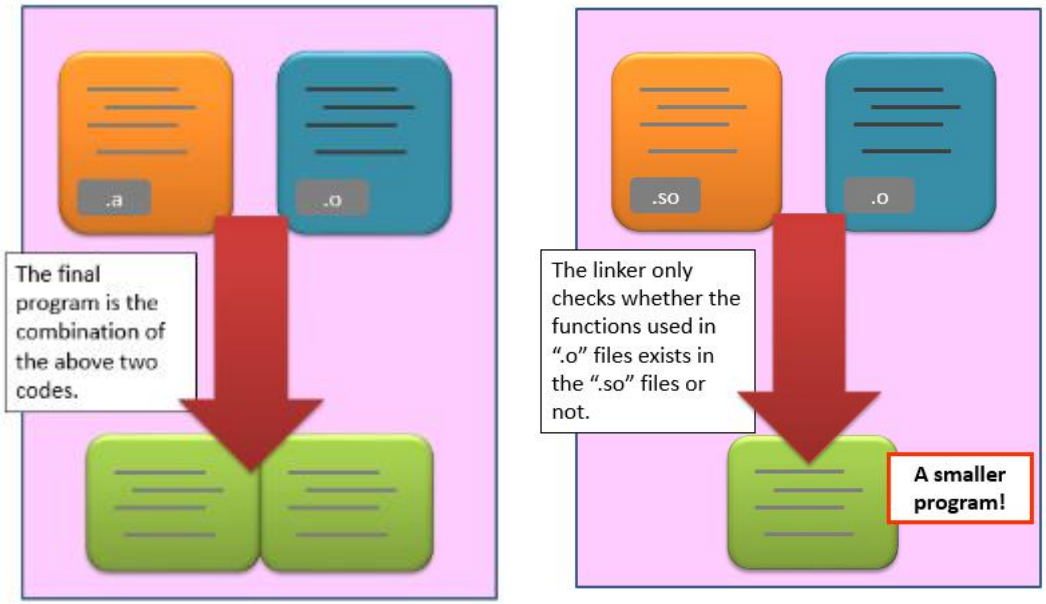
```
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    $0, -4(%rbp)
        addl    $1, -4(%rbp)
        addl    $1, -4(%rbp)
        addl    $1, -4(%rbp)
        movl    -4(%rbp), %eax
        movl    %eax, %esi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
```

opt1.s

```
main:
.LFB23:
        .cfi_startproc
        subq    $8, %rsp
        .cfi_def_cfa_offset 16
        movl    $3, %edx
        movl    $.LC0, %esi
        movl    $1, %edi
        movl    $0, %eax
        call    __printf_chk
        movl    $0, %eax
        addq    $8, %rsp
        .cfi_def_cfa_offset 8
        ret
        .cfi_endproc
.LFE23:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0
        .section        .note.GNU-stack,"",@progbits
```

WHAT HAPPENS?

- **Finally, Linker will link share library or static library with your code. And form executable file.**



- **Pictures from: https://calvinkam.github.io/csci3150-Fall17-lab3/assembler-and-linker.html**

C LANGUAGE

Pictures from google

WHY WE NEED C

编程语言排行榜 TOP 50 榜单

排名	编程语言	流行度	对比上月	年度明星语言
1	Java	15.876%	⌄ 1.028%	2015, 2005
2	C	12.424%	⌄ 0.913%	2017, 2008
3	Python	7.574%	⌄ 0.72%	2010, 2007, 2018
4	C++	7.444%	⌄ 0.714%	2003
5	Visual Basic .NET	7.095%	⌃ 0.636%	
6	JavaScript	2.848%	⌄ 0.454%	2014
7	C#	2.846%	⌄ 0.438%	
8	PHP	2.271%	⌄ 0.409%	2004
9	SQL	1.900%	⌄ 0.377%	
10	Objective-C	1.447%	⌄ 0.334%	2012, 2011

EXAMPLE

Please write a simple program.

Get 2 integer from user input

Print the square root of their square sum.

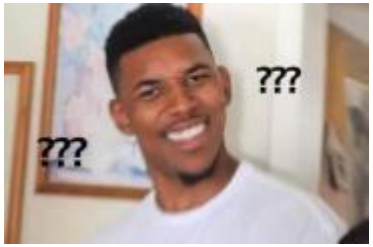
EXAMPLE

- **We write the code and save it. Do you remember how to use vim?**

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

int main(){
    int x, y;
    double res;
    scanf("%d%d", &x, &y);
    res = sqrt(pow(x, 2) + pow(y, 2));
    printf("%f\n", res);
}
```

- **Then compile it.**

```
mark@ubuntu:~/os_teaching$ gcc sqrt.c
/tmp/ccq9z1iW.o: In function `main':
sqrt.c:(.text+0x47): undefined reference to `pow'
sqrt.c:(.text+0x64): undefined reference to `pow'
sqrt.c:(.text+0x6e): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
```



EXAMPLE

- **Do not be afraid of meeting problems, we have many ways to solve it.**
- **Let's search it on the internet.**
- **https://stackoverflow.com/questions/13228111/c-undefined-reference-to-sqrt**
- **Here is the solution!**

you should link the math library when compiling

```
-lm
```

EXAMPLE

OOPs, it works. Thanks to stackoverflow!

- **Let's try again.**

```
mark@ubuntu:~/os_teaching$ gcc sqrt.c -lm
mark@ubuntu:~/os_teaching$
```

- **And it works!**

```
mark@ubuntu:~/os_teaching$ ./a.out
4
5
6.403124
mark@ubuntu:~/os_teaching$
```

STRANGE CODE

```c
#include<stdio.h>
int main(){
    int a[10];
    for(int i=0;i<=10;i++)
    {
        a[i] = 0;
//      printf("%p %p\n", &a[i], &i);
    }
}
```

Sometimes these code will cause infinite loop, sometimes will not. Why?

Try yourself

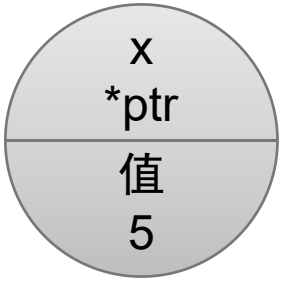
POINTER

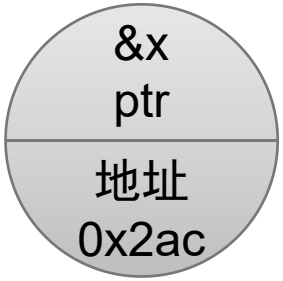
- 指针——C语言的精髓

```c
#include<stdio.h>

void add(int* x){
    (*x)++;
}

int main(){
    int x=0;
    add(&x);
    printf("%d\n",x);
}
```

& 取址操作符
* 取值操作符

```c
int x = 5;
int *ptr = &x;
```





You will meet pointer many times in this course.

EXERCISE

Pictures from google

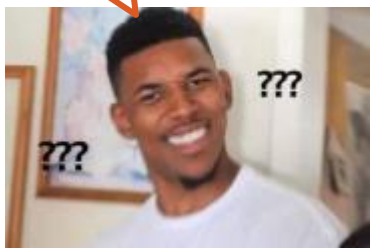
EXERCISE

- **gcc exercise**
  - Try gcc hello.c what do you find?
  - Try gcc –c hello.c what do you find?
  - Try gcc –E hello.c what do you find?
  - Try gcc –S hello.c what do you find?
  - How about add –o output to these commands?

EXERCISE

- **Be aware**
  - How to use array
  - The average number

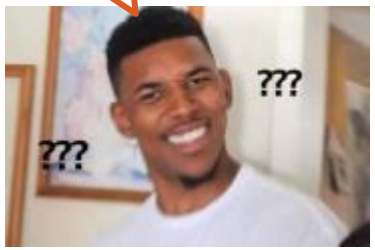
I will input 20 integers, please calculate the maximum, minimum and the average number of these 20 integers.



EXERCISE

- **Be aware**
  - How to sort them?
  - Can you use library?

I will input n integers,(1 <= n <= 100), please sort them by ascending order.

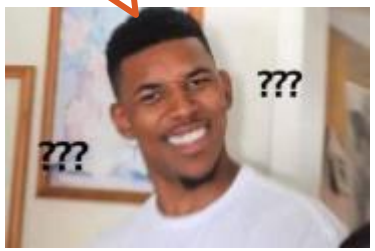


EXERCISE

- **Be aware**
  - spaces

Give you an integer n. Please print the following picture. (ex. n = 7)
n <= 11, n is odd.

```
  *
 ***
*****
*******
*****
 ***
  *
```

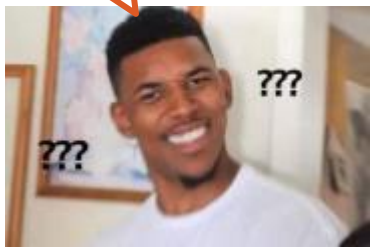


EXERCISE

- **Be aware**
  - directions

Give you an integer n. Please print the following picture. (ex. n = 9)
n <= 100, n is a square number

123
894
765



THANK YOU