ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

# PROGRAMMING METHODOLOGY
(PHƯƠNG PHÁP LẬP TRÌNH)

## UNIT 1: Computing Fundamentals

## Acknowledgement

- The contents of these slides have origin from School of Computing, National University of Singapore.
- We greatly appreciate support from Mr. Aaron Tan Tuck Choy for kindly sharing these materials.

## Policies for students

- These contents are only used for students PERSONALLY.
- Students are NOT allowed to modify or deliver these contents to anywhere or anyone for any purpose.

## Recording of modifications

- Currently, there are no modification on these contents.

## Unit 1: Computing Fundamentals

1. Hardware and Software
2. Program Development
3. Programming Environment
4. sunfire – a UNIX machine
5. vim – a text editor
6. File transfer

## Hardware

Monitor and speaker **(output)**

Houses processor, memory, buses, etc.

Keyboard and mouse **(input)**

## Software

- Set of instructions to perform tasks to specifications
- Programs are software

http://www.tutorialspoint.com/computer_fundamentals/computer_quick_guide.htm

- (Computer) Program
  - Sequence of instructions for a computer to execute
- Programming languages
  - Languages for writing programs

## Types of Programs

- Machine code

Program to which computer can respond directly. Each instruction is a binary code that corresponds to a native instruction.

Eg: 0001001101101110

- Assembly code

Low-level language with strong (generally one-to-one) correspondence between assembly code and machine code instructions.

Eg: MIPS (add t1, t2, t3)

*Requires translation*

- High-level language program

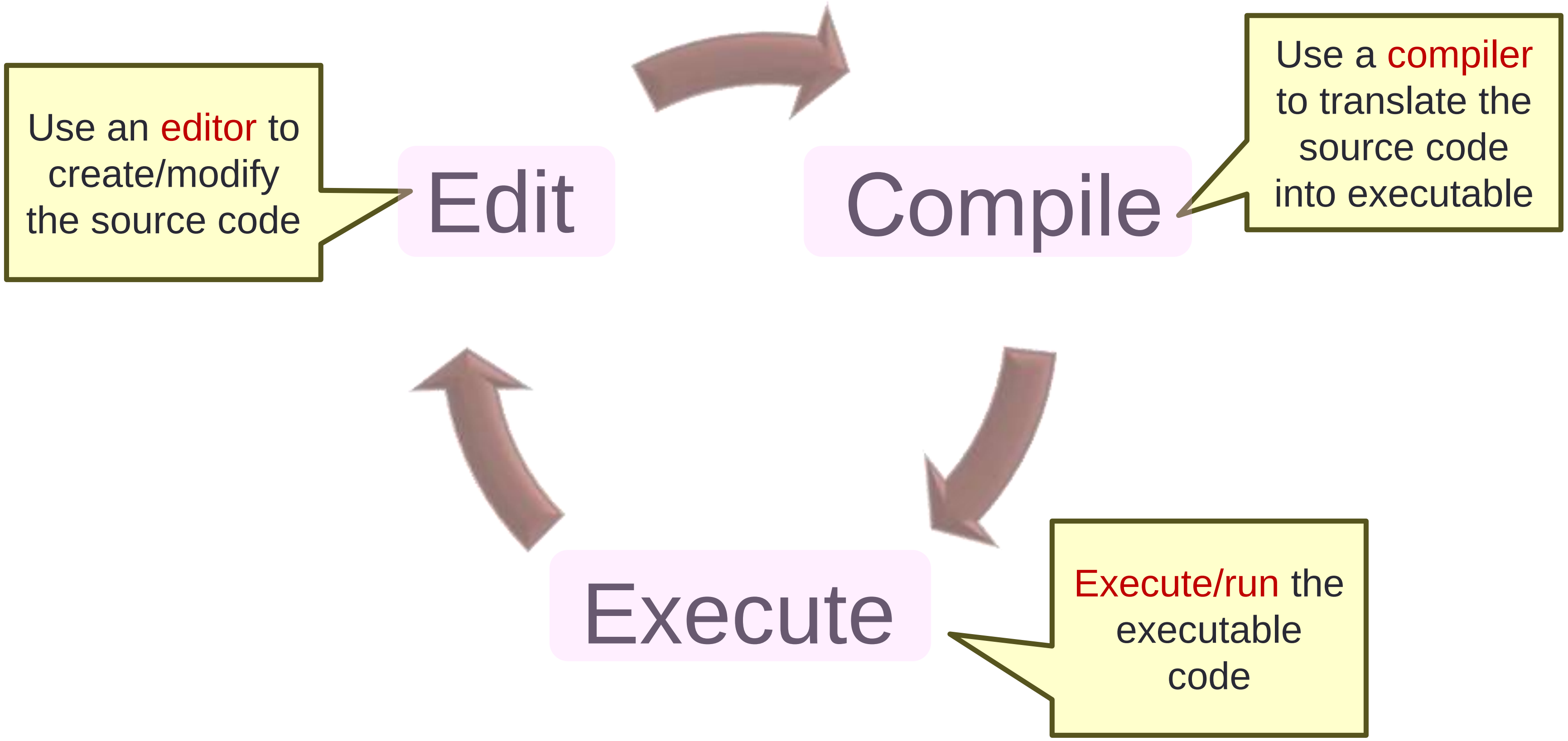
Detailed knowledge of the machine is not required. High level of abstraction. Ease of writing and understanding.

Eg: Java, C, C++, Python.

## Translation of Programs

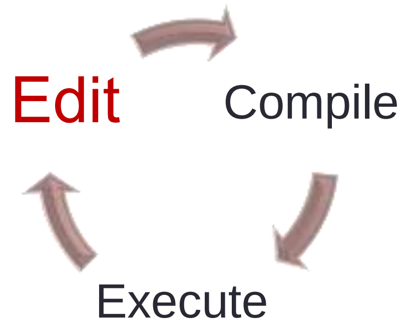
- High-level language programs (eg: C) cannot be executed directly by the computer
- Require a translation process called compilation
- A special program called compiler is used
- The original C program is called the source code
- The compiled program is the executable code or machine code
- In general, executable codes generated on a certain machine cannot be executed on another machine with a different architecture
  - The source code needs to be compiled on the new machine

## The Edit, Compile and Execute Cycle

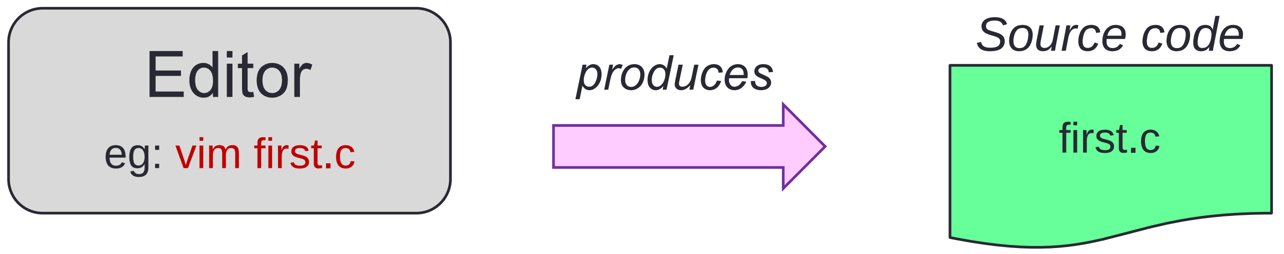

Use an editor to create/modify the source code
Edit
Compile
Use a compiler to translate the source code into executable
Execute
Execute/run the executable code

*Process is iterative*

## Editing C source codes (1/2)

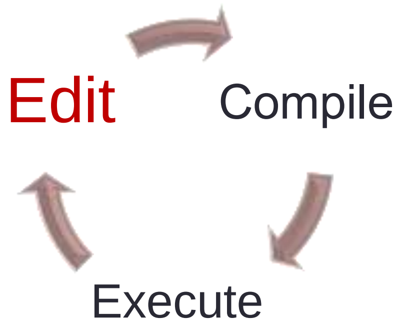


- We use a text editor to create/modify C programs (source codes)
- We will use the vim editor



- vim is a powerful text editor. It has 2 modes
  - Command mode: for issuing vim commands
  - Insert mode: for typing in text
- To switch between command mode and insert mode
  - Type i in command mode to get into insert mode
  - Press <esc> key in insert mode to get into command mode

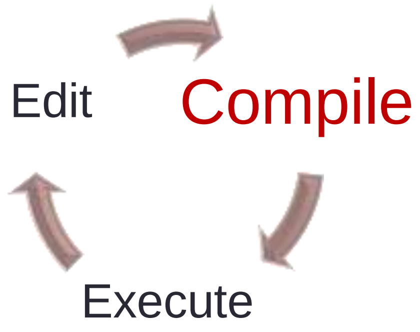
## Editing C source codes (2/2)



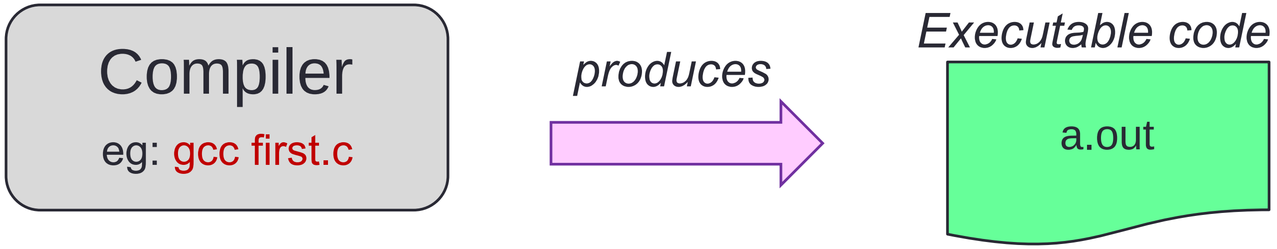
- Use vim to create this C program first.c

```
#include <stdio.h>

int main(void) {
   int a=27, b=6, c;

   c = a%b;
   printf("The value of c is %d.\n", c);

   return 0;
}
```

# Compiling C programs (1/3)



- We use the C compiler gcc in sunfire

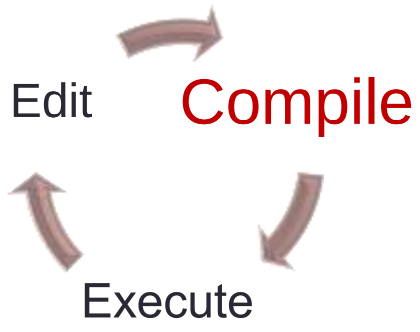


- Advisable to add the option –Wall (warnings all) for beginners:

```
gcc –Wall first.c
```

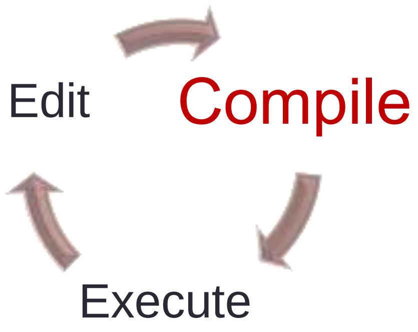
- If there are compilation errors/warnings, you need to edit the source code first.c again (vim first.c), and re-compile (gcc –Wall first.c), until your code is clear of compilation errors/warnings.
- Remember to add option '-lm' if your C program uses math functions
  - Example: `gcc –Wall –lm example1.c`
- Type 'ls' to check that you have the executable code a.out

## Compiling C programs (2/3)



- The executable file has the default name a.out. However, all filenames in a directory must be unique, hence there can only be one a.out in a directory.
- Since you have many C source codes in a directory (eg: example1.c, example2.c, example3.c), you might want to have their corresponding executable files all in the same directory, appropriately named.
- Two approaches:
  1. Rename a.out after compilation
  2. Indicate the desired name of the executable file during compilation

# Compiling C programs (3/3)



1. Rename a.out after compilation

```
happytan@sunfire [] ~/c $ gcc -Wall -lm example1.c
happytan@sunfire [] ~/c $ mv a.out example1
happytan@sunfire [] ~/c $ gcc -Wall example2.c
happytan@sunfire [] ~/c $ mv a.out example2
happytan@sunfire [] ~/c $ gcc -Wall example3.c
happytan@sunfire [] ~/c $ mv a.out example3
```

Executable files are named example1, example2, example3.

2. Indicate the desired name of the executable file during compilation using the '-o' option

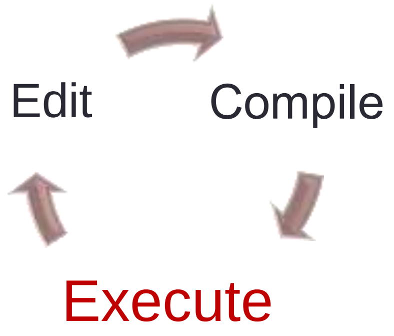
```
happytan@sunfire [] ~/c $ gcc -Wall -lm example1.c -o example1
happytan@sunfire [] ~/c $ gcc -Wall example2.c -o example2
happytan@sunfire [] ~/c $ gcc -Wall example3.c -o example3
```

Be careful not to overwrite the source code accidentally!
The following will replace the source code with the executable file, which is called example1.c now! The source code cannot be recovered!

WRONG WAY

```
happytan@sunfire [] ~/c $ gcc -Wall -lm example1.c -o example1.c
```

# Executing C programs

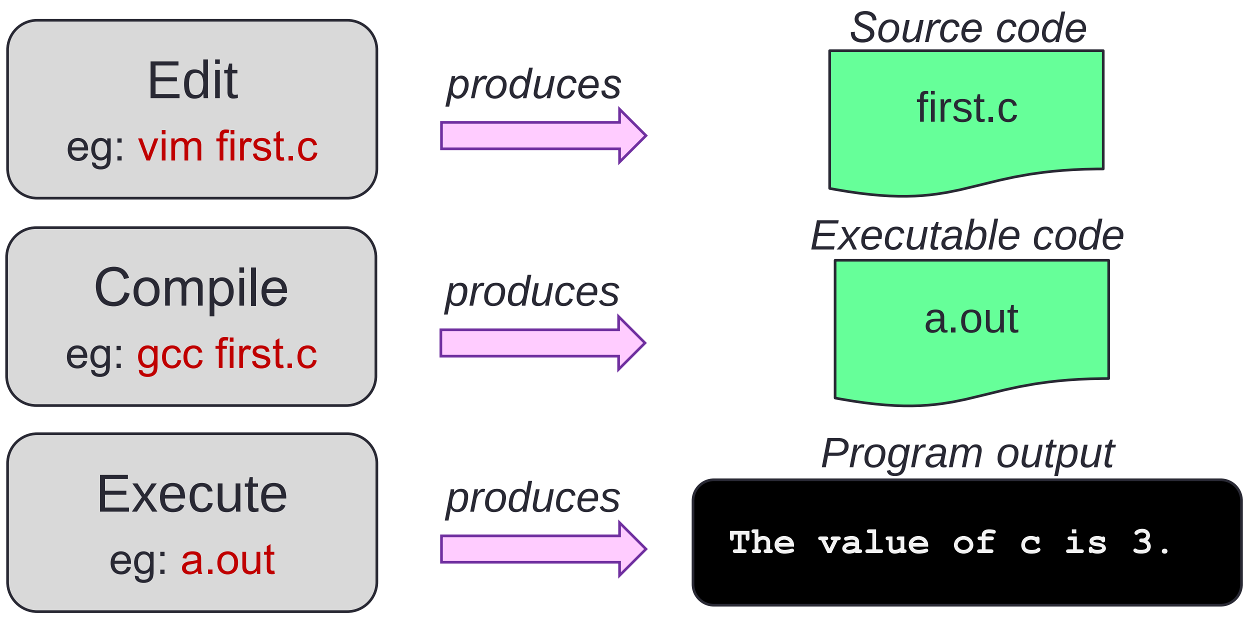


- Executing a C program is simple – just type the name of the executable file

To run the executable file example1:

```
happytan@sunfire [] ~/c $ example1
The distance between the 2 points is  3.61
```

- We have gone through the Edit – Compile – Execute process



```
The value of c is 3.
```

## Summary

- In this unit, you have
  - Familiarised yourself with the programming environment
  - Accessed the sunfire system and learned some basic UNIX commands
  - Used the editor vim to create/modify your C programs
  - Used the compiler gcc to compile your C programs
  - Familiarised yourself with the edit – compile – execute process

End of File