Tutorial 1 Warm Up!

TA: HOU Weiying
Email: wyhou@cs.hku.hk

TA: HUANG Songlin
Email: huangs0@connect.hku.hk

STA: Kong Chun Yung
Help QA on moodle!

Objectives

PART 1

- Environment Setup
  - Why we need Linux?
  - Local: To write & debug your code
    - [video] MacOS user
    - [video] Windows user
    - Linux user
  - Remote: Work & study everywhere
    - ~~Guacamole~~
    - Termius
- Lab-0: Welcome to Linux
- Lab-2: Signal

PART 2

- Useful C Language Skills in this course
  - Review: A simple C file
  - Pointer:
  - Function/Procedures: Passing Parameters
  - Manually Allocate memory
  - Compile (GCC)
  - Debug
- Lab-1: C Basics

Environment Setup

- Why Linux?
  - Linux is one of the most **widely-used** around the world, and most importantly it is **open-source**.
  - We use **workbench2** for grading, and it is Ubuntu system, a user-friendly Linux distribution.

Linus Torvalds

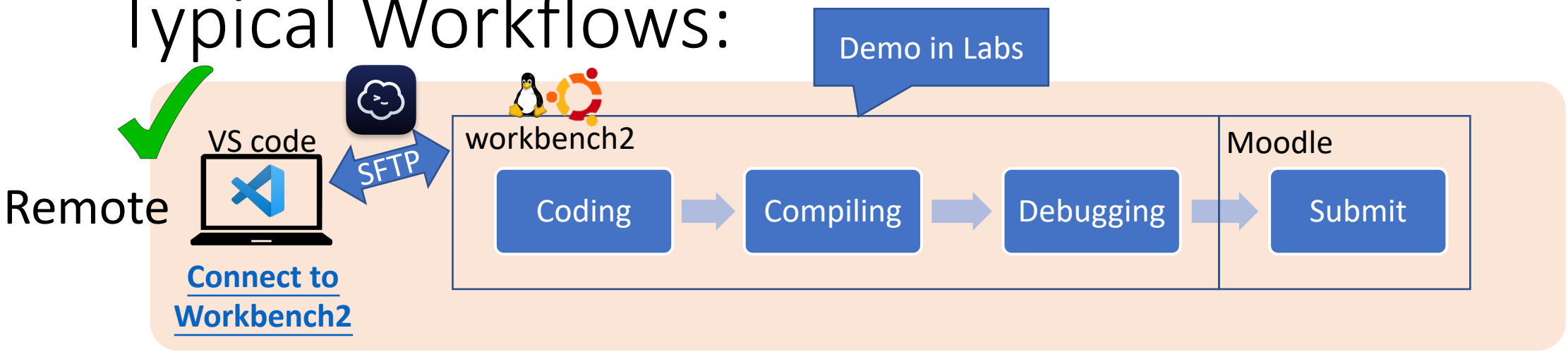
Wrote first version of

Linux

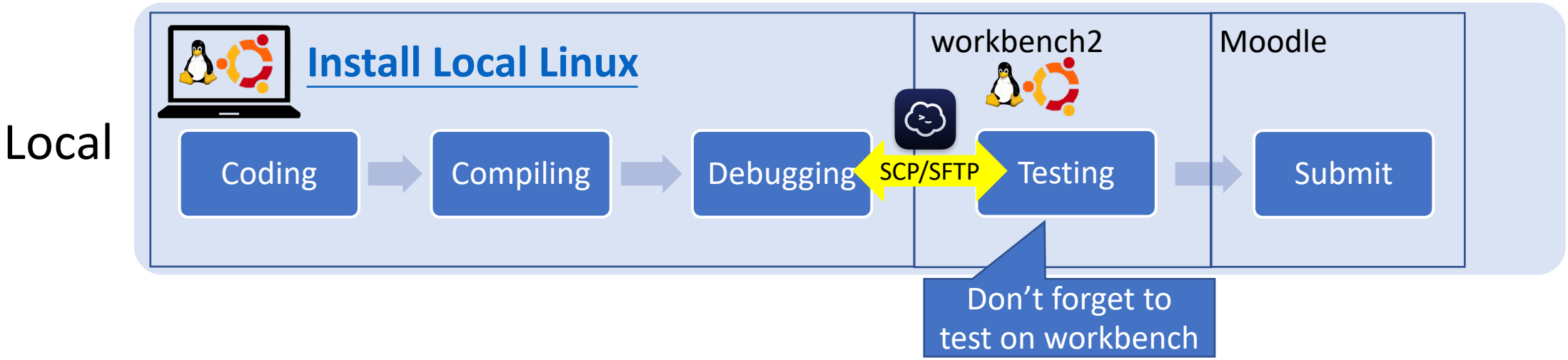
Has a popular distribution named

ubuntu

Typical Workflows:



For future Programming Assignments



Recommended for everyone

Remote: Connect to Workbench2

Three Linux servers available for us: academy11, academy21, **workbench2(grading)**

- Apache **Guacamole:** Moodle > Computing Platform Section (not recommended)

- *Termius*: cross-platform SSH client (recommended)
  - First, connect to HKUVPN or SSH to gatekeeper.cs.hku.hk
  - Then, workbench2.cs.hku.hk

- *VS code*: powerful and open-source text editor (recommended)
  - Setting is similar to Termius, but use SSH configuration file

Typical Workflows

Only for Pro Windows/macOS user

Local: Install Local Linux [videos]

- Windows Users
  - WSL: Windows Subsystem for Linux
  - Docker
- macOS Users
  - Docker, ~~Clang, gcc-9~~
- Linux user:
  - better use gcc-9.4

To avoid trouble. Do not compile and debug on macOS!

Lab-0: Get started with Linux Command-line

- man [command]
  - Open the manual pages of the command.
- ps [option] -- shows **p**rocess **s**tatus
  - -e Select all processes
  - -f Show in full format
  - w Wide output
  - f ASCII-art process hierarchy (forest)
- pstree -- shows relations between processes
- top -- display Linux processes

Lab-0: Get started with Linux Command-line

- Linux commands
  - Basic: Linux commandline for beginners
    - |: pipe output of one cmd to another
    - > and >>: Output redirection. >: create new file; >>: append to existing file
    - &: run in background
    - …

Let's do some coding

- Download from
  - Github repo: (link on the right)
    - Git clone (recommended)
    - Download as zip file
- materials at:
  - Tutorial1-Lab1-signal/*.c
  - Tutorial1-Lab2-virtual_memory/*.c

- For git beginners: [More details]
  1. git is installed by default on Ubuntu.
     - Otherwise, run sudo apt install git
  2. Run git clone https://github.com/aiot-lab/HKU-COMP3230A-Tutorialabs.git in your command-line. It will download from the Github the code repository.
  3. Get updates by running git pull in the directory `HKU-COMP3230A-Tutorialabs`
  4. more git command. Check this link

Write A simple C program on the remote server

- Steps
  - 1. Connect to workbench2.cs.hku.hk
    - via termius <-- a good way to get access to command-line + SFTP
    - via VS code (or other editor)
  - 2. Use any text editor to write a simple C file
    - (local) write -> upload to remote -> compile on workbench2
    - (remote, command-line) vim
    - (remote) VS code <-- recommended
  - 3. Use GCC to compile & run as an executable file
- more examples you can play with in the git repo~

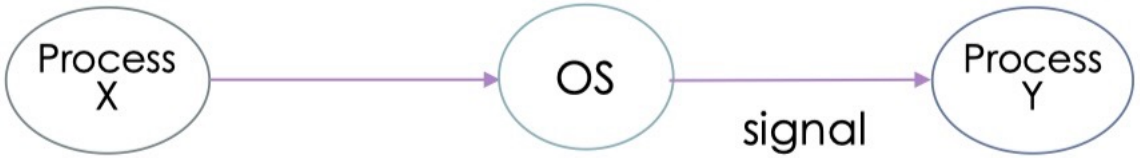
C File Compile

- GCC: Tutorial
  - A cross-platform compiler for C Language
  - Example: ./C-basic/01-gcc-basic.c

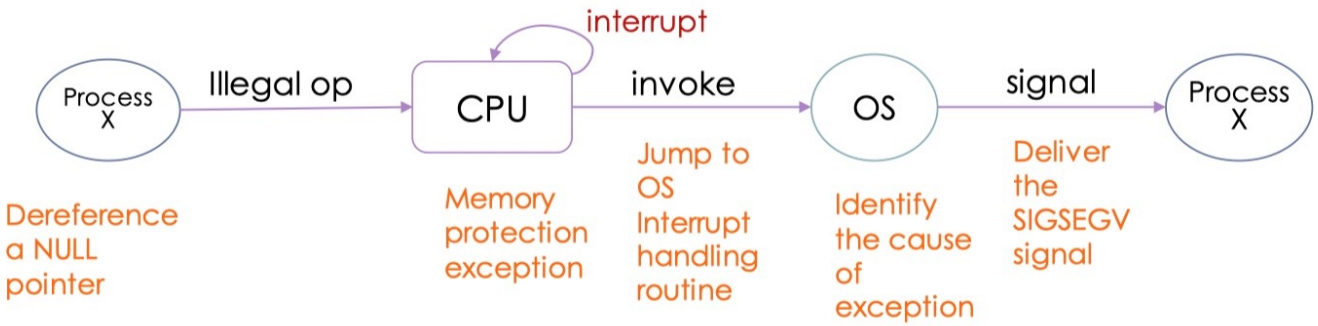
- Makefile: Tutorial
  - A list of rules to execute commands
  - Only compile what was changed
  - Example: ./Tutorial1-Lab1-signal/makefile

Lab-2: Signal

- Review: Signals are used in UNIX systems to notify a process that a particular event has occurred.
  - Synchronous signal
  - Asynchronous signal



- In <signal.h>, the function signal(int sig , void (*handler)(int)) is used to specify which routine to handle the incoming signal from the OS.



Lab-2: Signal

Q1: What will happen if we override default system handler for SIGFPE and then trigger div0 exception? Try to explain why you succeed or fail. You can compare with signaling SIGFPE with command kill -8 <pid>.

- kill -8 <pid> can be handled by

```
void signal_handler(int signum) {
    printf("Caught signal %d (%s, %s)\n", signum, signal_list[signum], signal_des[signum]);
}
```

- While div0 exception cannot… why?

ALU(Arithmetic logic unit) flags have been changed while div0! Use setjmp and longjmp to store and recover register flags.

The END of part1

Have a break

Objectives

PART 1

- Environment Setup
  - Why we need Linux?
  - Local: To write & debug your code
    - [video] MacOS user
    - [video] Windows user
    - Linux user
  - Remote: Work & study everywhere
    - ~~Guacamole~~
    - Termius
- Lab-0: Welcome to Linux
- Lab-2: Signal

PART 2

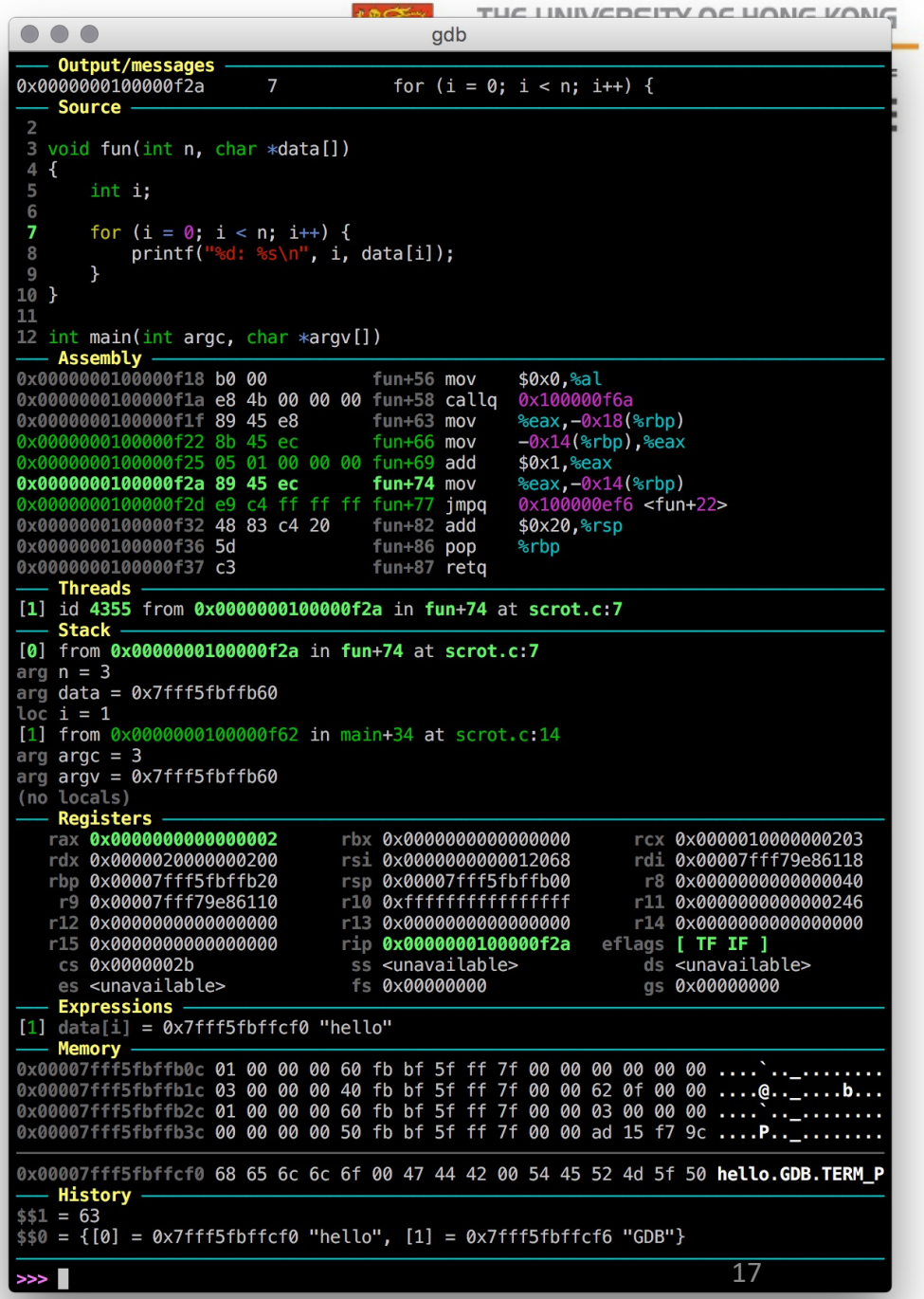
- Useful C Language Skills in this course
  - Review: A simple C file
  - Pointer:
  - Function/Procedures: Passing Parameters
  - Manually Allocate memory
  - Compile (GCC)
  - Debug
- Lab-1: C Basics

Lab-1: C Basics …

- Basic data type: Variable, Pointer, ...
- Data structure: Linked list, matrix…
- Parameter passing: by value, by reference, by address …
- Preprocessor and Macro
- recursive function calls
- malloc/free…
- References
  - Textbook&projects of the previous C/C++ programming courses you finished
  - Look up C Lang's syntax online:

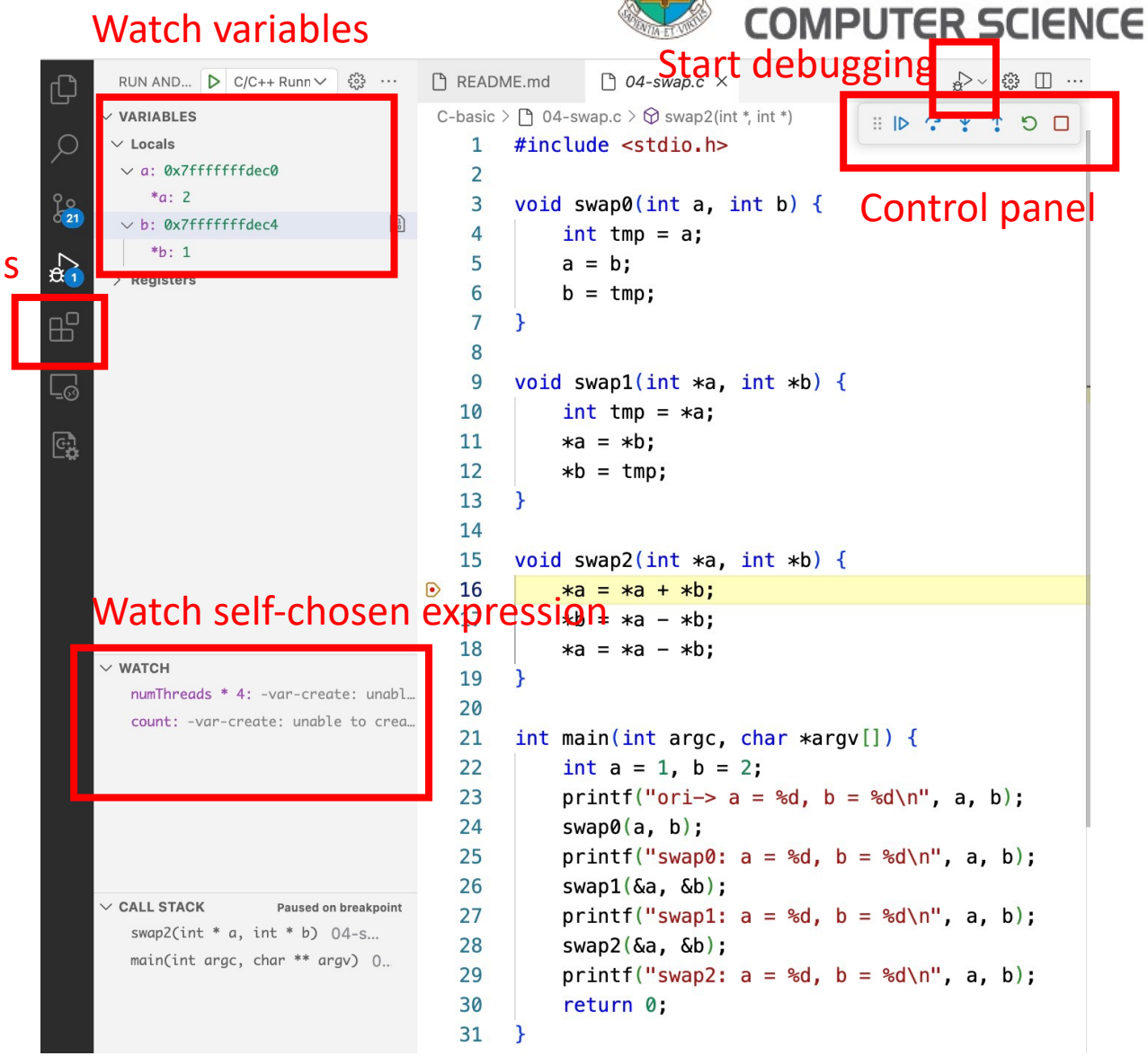
Debug: GDB

- gdb (GNU Debugger) is a program that runs another program and allows you to track the execution.
- Official Documents
- Basic Tutorial
- Basic usage:
  - Compile for debug, add flag `–g`
  - gdb <file>
- GDB config (with highlight): Tutorial
- GDB Dashboard



Debug: VS Code

- Step:
  1. Install C/C++ Extension Pack and C/C++ Runner
  2. Let's debug on one click!



The END

HOU Weiying
Email: wyhou@cs.hku.hk