

CS110 Discussion

Week2

Guanghui Hu

February 24, 2025

content

- 1 Linux and CLI
- 2 Git
- 3 GCC and GDB
- 4 Static and dynamic library
- 5 Makefile

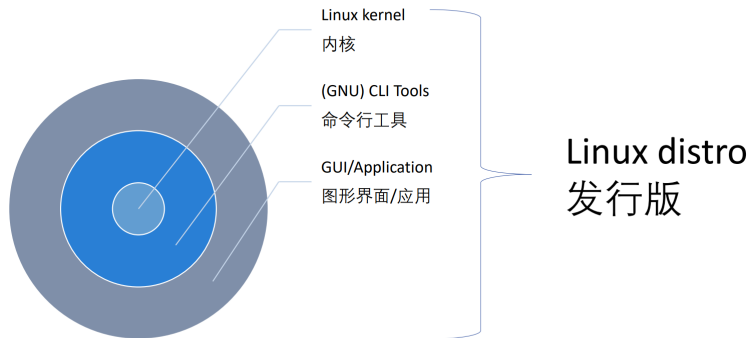
preface

- Not just a discussion, more like an **Lab**.
- Lots of **practical content**, a little hard for Linux beginners.
- There will be **code and command line operations**.



What is Linux

- Linux is a family of open-source Unix-like **operating systems** based on the Linux kernel.
- There are lots of Linux distributions such as Ubuntu, Debian, Arch, NixOS...



Install a Linux system

- Opt1: dual system
- Opt2: virtual machine(a in-class demo)
 - Easy to start kubuntu20.04, 22.04 virtual machine: [Linux 101](#)
 - ubuntu20.04, 22.04, 24.04 image: [Tsinghua mirrors](#)
 - VirtualBox vs VMware
- WSL is not suggested

Shell and terminal

- **Shell:** A family of programs which enable user to interact with kernel, like bash, csh, zsh...
- **Terminal:** A TUI/GUI where user can type in their command. Each time a terminal is created, it will run shell.
 - **Gnome Terminal:** enter `ctrl+alt+T` to open gnome default GUI terminal
 - **TTY:** enter `ctrl+alt+F1` to open TUI terminal

Basic usage of Shell

- **whoami**: print current user name.
- **hostname**: print this machine's name.
- **cd** [*directory*]: switch to directory
- **ls**: list all the files and directories under current directory.
- **pwd**: print absolute path of current directory.
- **mkdir** [*name*] : create an empty under current directory.
- **rm** [*file..*] : delete files.
- **sudo apt install** [*software name*]: install commands with superuser privileges.
- **man** [*command*]: show manual of the command.
- **touch** [*file*]: change file timestamps.
- **cat** [*file*]: print the content of file.
- **vim** [*file*]: terminal editor.
- **mv** [*old name*] [*new name*]: move or rename file or directory.

Shell script

- Shell script is a file written by commands and shell syntax.
- The shell will interrupt and execute it

```
#!/bin/bash
mkdir hello
cd hello
for i in {0..9}
do
    touch "hello$i.c"
done
```


Reference about shell syntax and script

- [Missing Semester of Your CS Education](#): MIT's beginner-oriented Linux courses, **highly recommended!**
- [Linux 101 chapter9](#): Introduction to useful command line tools.
- [Bash scripting cheatsheet](#): A table of common Bash syntax

- 1 Linux and CLI
- 2 Git**
- 3 GCC and GDB
- 4 Static and dynamic library
- 5 Makefile

Introduction of Git

Git is a **distributed version control** system that tracks changes in any set of computer files, usually **used for coordinating work** among programmers collaboratively developing source code during software development.

In the beginning

- install Git in linux : `sudo apt install -y git`
- Initialization configuration :

- by command line :

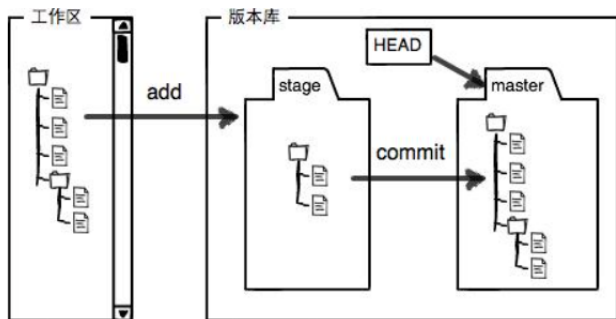
```
git config --global user.email "email@example.com"  
git config --global user.name "Your Name"
```

- by configuration file :
create file `~/.gitconfig` or `~/.config/git/config`
add below information

```
[user]  
    email = "email@example.com"  
    name = "Your Name"
```

Three important areas of Git

- 1 **workspace:** a folder in your computer which stores all files of your project
- 2 **stage:** a temporary area which is used to store your changes
- 3 **repository:** the place where all the information and files are stored



How to create a Git repository

- 1 use **git init** under a folder to create an empty repository, then the folder will be a git repository.
- 2 use **git clone *url*** to download an existed repository,
 - *url* could be a website or a local path

How to manage a local Git repository

After create or clone, a Git repository is now in your computer. The following command is the most frequently used to manage your Git repository

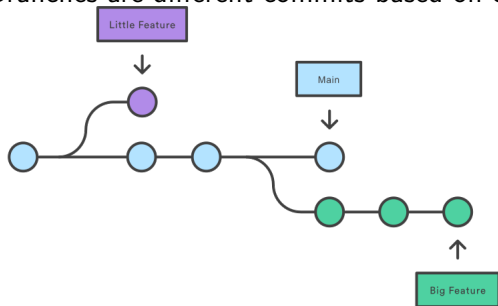
- 1 use **git add [files..]** to add some files from **workspace** into the **stage**
- 2 use **git commit [files..]** to add files from **stage** into **repository**
- 3 use **git status** to check the current status of the git repository
- 4 use **git log** to view all the previous commit
- 5 use **git checkout - - [files..]** to discard changes in workspace
- 6 more in [official document](#)

How to manage a remote Git repository

- 1 use **git remote -v** show remote repository details *name*
- 2 use **git pull** to update the local git repository
- 3 use **git push** to push local commit at *branch* to the remote repository

What is a branch

Branches are different commits based on one commit.



- use **git branch** to check all the branches
- use **git checkout -b [name]** to create branch
- use **git checkout [name]** to switch branch
- use **git merge [name]** to merge changes into the current branch.

Reference about Git

- ① [MIT Git tutorial](#): Introduction to basic Git concepts and common commands with video.
- ② [Git document](#): long but useful for search.
- ③ [Liao Xuefeng's tutorial](#): Chinese Git introduction with video.

- 1 Linux and CLI
- 2 Git
- 3 GCC and GDB**
- 4 Static and dynamic library
- 5 Makefile

Introduction of GCC

- GCC is short for GNU Compiler Collection. It is a widely used compiler for C and C++.
- Use GCC to compile C and g++ for C++.
- Basic command: `gcc xxx.c -o xxx`
- you can add arguments to enable some functions

Warning operation

- **-Wpedantic**: enable warnings about compiler or language compatibility.

```
// gcc -std=c99 -Wpedantic test.c
// warning: ISO C forbids braced-groups within expressions
int main() {
    int a = ({ int b = 5; b; });
    return a;
}
```

- **-Wall**: enables warnings about constructions that some users consider questionable, and that are easy to avoid.
- **-Wextra**: enables extra warning that are not enabled by -Wall
- **-Werror**: Make all warnings into errors.
- more to see [Options to Request or Suppress Warnings](#)

Optimization operation

- **-O0,-O1,-O2,-O3** : different level of optimization.
- **-flto** : link-time optimizer.
- more to see [Options That Control Optimization](#).

Some other useful arguments

- **-I**: specify path for finding header.
- **-std** : specify the compile standard.
- **-g** : add debug information into the executable files.
 - optimization option **-Og** is recommand if **-g** is used.

Introduction of GDB

- GDB is short for GNU Debugger.
- Add **-g** option to enable GDB, when compiling the source code.
- To start GDB, use **`gdb file`** , where file is the object file.

Basic usage of GDB

- **h** [*args.*]: print help for command *args*.
- **r** [*args.*]: start the program with *args*.
- **b** [*line number*] : set breakpoint.
- **c**: Continue program being debugged, after signal or breakpoint.
- **s**: Step program until it reaches a different source line.
- **n**: Step program, proceeding through subroutine calls.
- **p** [*vars*]: print variable.
- **layout** [*vars*]: change the layout of windows.
- **bt** Print backtrace of all stack frames.
- **f** Select and print a stack frame.

Reference for GCC and GDB

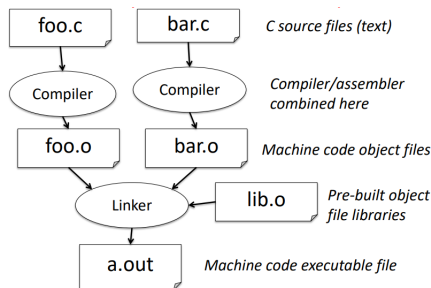
- [Unix Programming Tools](#): a short but practical introduction about GCC and GDB.
- [Stanford's GDB and Debugging](#): Beginner's guide to GDB with video.
- [GDB quick reference](#): cheatsheet about GDB commands.

- 1 Linux and CLI
- 2 Git
- 3 GCC and GDB
- 4 Static and dynamic library**
- 5 Makefile

Link

Building process

- assemble: c syntax text code \rightarrow assemble code.
- compile: assemble code \rightarrow Executable and Linkable Format(ELF).
- **link: combines multiple ELF file into executable file or lib file.**



library

Libraries consists of **executable code** such as compiled functions and classes, and optional a collection of **header file**.

- Header file: such as `stdio.h`, used by developers to write code.
- Static library (executable code):
 - File with suffix `.a`
 - Totally integrated into executable programs.
 - Program can run without static library.
- Dynamic library (executable code):
 - File with suffix `.so`
 - Only integrated little symbolic information to executable program.
 - Program will look for the required symbols(function, variable) **when it is running**.
 - Program **can not** run without dynamic library.

Static library and static linking

Commands related to static linking

```
gcc -c <lib source> -o <lib object> # compile library
ar -rs <lib name> <lib object>      # generate library
gcc <source use lib> <lib name>     # use library
```

Advantage:

- fast
- program can run independently

Disadvantage:

- large files waste the memory
- combines all ELF together, even those not used code.

Dynamic library and dynamic linking

Commands related to dynamic linking

```
gcc -fpic -c <lib source> -o <lib object> # compile library
gcc -shared -o <lib name> <lib object> # generate library
gcc <source use lib> <lib name> # use library
```

Advantage:

- load needed modules when running, do not need to link all files together before running.
- easy to maintain separately
- decrease the size of executable files

Disadvantage:

- more complex
- may influence the performance

Useful commands about Link

- **nm**: List symbol names in object file
- **readelf**: Parser and display information about ELF files
- **ldd**: Display shared library dependencies of a binary.

```
ubuntu@cloud-img:~$ ldd /usr/bin/gcc  
linux-vdso.so.1 (0x00007ffcd9fcd000)  
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fd11b5be000)  
/lib64/ld-linux-x86-64.so.2 (0x00007fd11b7f7000)
```
- **objdump**: View information about object files

Reference about link and static/dynamic libraries

- Classic Textbooks: **Computer Systems: A Programmer's Perspective**(CSAPP) chapter7 Linking

图书

深入理解计算机系统 == Computer systems : a programmer's perspective

布赖恩特, (Bryant, Randal E.) 奥哈洛伦, (O'Hallaron, David R.) 龚奕利 雷迎春

北京 : 机械工业出版社 2011

 在架 上科大图书馆 404专业阅览室 (TP3/4) >

- CMU lecture slides for CSAPP chapter7 Linking

- 1 Linux and CLI
- 2 Git
- 3 GCC and GDB
- 4 Static and dynamic library
- 5 Makefile**

Makefile introduction

What can a Makefile Do:

- **Reduce Repetitive Tasks:** Avoid manually typing long compiler commands for every file.
- **Manage Dependencies:** Automatically compile only the parts of the project that have changed.

What is Makefile:

- A special **file** used by the CLI **make** to automate some process.
- It defines **a set of rules** for how to compile and link programs.
- Each rule contains **target, dependencies and build commands**.

Makefile's rule

Makefile Rule Syntax:

```
target: dependencies...  
    build commands...  
...
```

- **target:** File name or target name.
- **dependencies:** Files need for building target, can be empty.
- **build commands:** Extended shell scripts, could be multiple line.

How to use Makefile

- 1 create file named **Makefile** under any directory
- 2 write your rules in Makefile
- 3 type `make <target name>` in this directory

```
ubuntu@cloud-img:/tmp/make-example$ cat Makefile
hello:
    echo hello Makefile!
ubuntu@cloud-img:/tmp/make-example$ make hello
echo hello Makefile!
hello Makefile!
```

Practical tips

- build commands must **start with one tab**, not space
- declare a non-filename target with keyword `.PHONY`
- type `make -n <target>` can only print commands but not execute

Makefile's variable

Basic Usage:

- The default type of makefile variables is **string**
- define a variable with `VarName := value`
- use a variable start with `$(VarName)`
- Variables can appear in **build commands** or **create new variables**.

Little Tips:

- define a variable with `VarName ?= value` to set a default value.
- shell variable can be used in Makefile

Example: [branch var of cs110-discuss2](#)

Reference about Makefile

- [how-to-write-makefile](#): Chinese Makefile guide for beginners
- [Learn Makefiles With the tastiest examples](#): English Makefile guide for beginners with video.
- [Makefile manual](#): long but useful for search.

summarize

context:

- Linux system install and basic command line tools usage.
- Common Git commands: add, commit, log, status, clone, pull, push.
- GCC and GDB: how to compile, run and debug C in Linux.
- Brief introduction about static/dynamic link and library.
- Practical Makefile guidance.
- Demonstrate with [a Git repository](#) which include all topics.

Hope:

- Read the friendly manual.
- Search the friendly website.
- Ask the friendly LLM.