CS110 Discussion

Week2

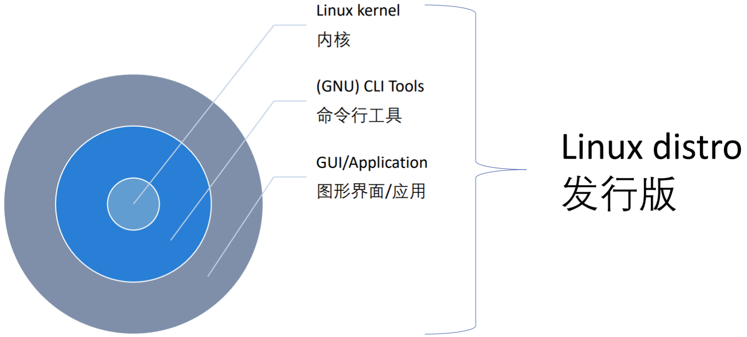
Guanghui Hu

February 24, 2025

content

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
  - VMware workstation player
- 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

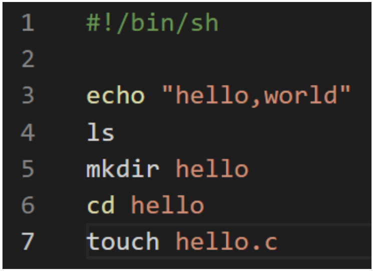
Basic usage of Shell

Some frequently used command

- **ls** : list all the files and directories under current directory
- **mkdir [*name*]** : create an empty under current directory
- **rm [*file..*]** : delete files
- **sudo** : do some thing with superuser privileges
- **apt install** : install commands using apt
- **man [*command*]** : show manual of the command
    - **tldr [*command*]** : show the simplified manual of a command

Shell script

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



```sh
#!/bin/sh

echo "hello,world"
ls
mkdir hello
cd hello
touch hello.c
```

- Please learn more from Missing Semester of Your CS Education

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 :

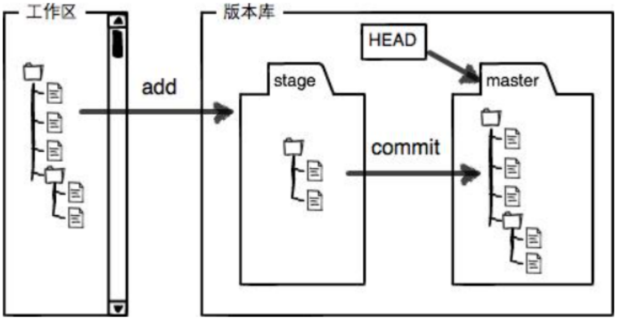
    ```
    git config --global user.email "email@example.com"
    git config --global user.name "Your Name"
    ```

  - 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. If you want to create an empty repository, you can use **git init** under a folder. Then, the folder will be a git repository.
2. If you want to download an existed repository, you can use **git clone *url***.

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

What is a branch

Branch is a special feature of git. With branches, one can have several works in progress at the same time and all of them are independent. Also, you can merge two branches.

- 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.

How to manage a remote git repository

1. use **git remote add [*name*] [*url*]** to add a remote repository through the *url* and name it as *name*
2. use **git pull [*name*] [*branch*]** to update the local git repository
3. use **git push [*name*] [*branch*]** to push local commit at *branch* to the remote repository

Some reference for you

1. Official document
2. MIT git tutorial
3. Liao Xuefeng's tutorial

Introduction of GCC

- GCC is short for GNU Compiler collection, which is a widely used compiler for C and C++.
- We will only use the C compiler(gcc) and C++ compiler(g++)
- Basic command: **gcc xxx.c -o xxx.o**
- you can add arguments to enable some functions

Warning operation

- **-Wpedantic** : Issue all the warnings demanded by strict ISO C and ISO C++
- **-Wall** : This enables all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros
- **-Wextra** : This enables some extra warning flags 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

- **-std** : specify the compile standard
- **-g** : add debug information into the executable files

Introduction of GDB

- GDB is short for GNU Debugger
- To enable GDB, you should add **-g** flag when compiling the source code.
- To start GDB, use **gdb** ***file*** , where file is the object file

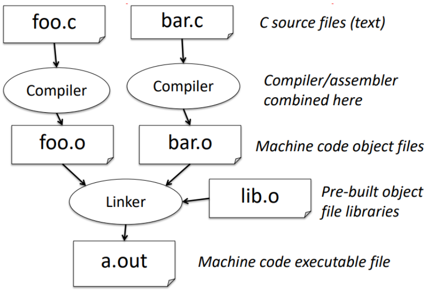
Basic usage of GDB

- **h** [*args..*] : print help for command args.
- **r** [*args..*] : start the program with args.
- **b** [*line number*] : set breakpoint
- **c** : continue running the program until the next breakpoint or error
- **s** : Runs the next line of program. If the next line is a function, step in it.
- **n** : Runs the next line of program. If the next line is a function, do not step in it.
- **p** [*vars*] : print variable
- **layout xxx** : change the layout of windows.

a GDB quick reference can be seen here

Link

During the process of compiling, we will first object files from the source files. Then we need to link the files with some library since we use library functions in the source files. There are two kinds of libraries: static library and dynamic library.


foo.c
bar.c
C source files (text)
Compiler
Compiler
Compiler/assembler combined here
foo.o
bar.o
Machine code object files
Linker
lib.o
Pre-built object file libraries
a.out
Machine code executable file

Static library and static linking

Static linking will link all the files and library together before running, even those not used code will be linked into the executable file.

- Advantage :
  - fast
  - program can run independently
- Disadvantage :
  - large files waste the memory

Dynamic library and dynamic linking

Dynamic linking is a technology that can load needed modules when running. Thus we do not need to link all files together before running.

- Advantage :
  - easy to maintain separately
  - decrease the size of executable files
- Disadvantage :
  - more complex
  - may influence the performance

Create library

- use **gcc** to create object files
- use **ar**/**gcc** to create libraries
- use **ld** to link
- to know more about these commands, see the manual or use command **man** .

Some useful commands

- **nm** : List symbol names in object file
- **ldd** : Display shared library dependencies of a binary.
- **objdump** : View information about object files

CMake introduction

CMake is cross-platform free and open-source software for build automation, testing, packaging and installation of software by using a compiler-independent method

Basic usage of CMake

- cmake_minimum_required(VERSION xxx): set the minimum required version of CMake
- PROJECT(xxx): name for the whole project
- SET(A B): assign B to A
- include_directories(xxx): set the including directories
- add_executable(A B C): build executable file A using B and C
- more to see Official CMake Document

Compile files with CMake

- write **CMakeLists.txt**
- make a directory **build**
- under **build**, run **cmake ..** and **make**
- the executable file is created in directory **build**
- a simple in-class demo