VE280 Recitation Class 1

Linux Operating System Introduction

Mao Junxiong

2016-5-30

Linux & Unix

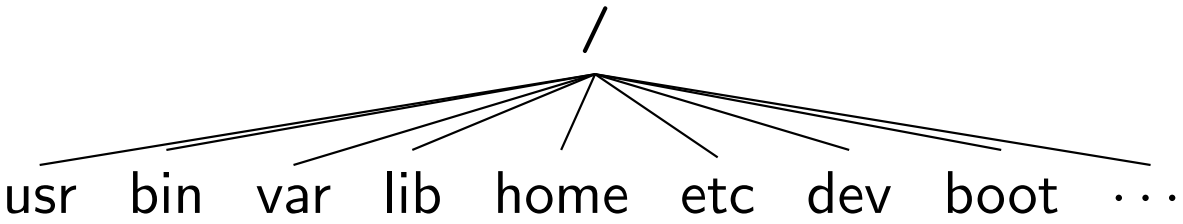
- Linux is a free and open-source Unix-like operating system, which was released by Linus Torvalds on October 5th, 1991.
- Unix is a family of multitasking, multiuser computer operating systems that derive from the original AT&T Unix, developed in the 1970s at the Bell Labs research center [1].

Why Linux?

- Free & open source: You can get the free operating system and source code from Internet. You can also study the code, modify it, and even redistribute it.
- Easy to install applications and environment support: sudo apt-get install.
- Large and multiple open-source communities.
- Stable.
- Etc...

Linux file system structure

- In linux, files are organized as a tree structure.



- Directories:
  - / Root directory.
  - . Current directory.
  - ∼ Current user home directory.
  - .. Parent level directory.

Basic commands in Linux terminal

- `cd`: Change your current working directory.
  - `cd /`: Change to root directory.
  - `cd ..`: Change to parent directory.
  - `cd newdir`: Change to the directory (named `newdir`) under the current directory.
- `ls`: List all files in the current directory.
  - `ls -l`: List all files under current directory in long format.
  - `ls /usr`: List all files under directory `/usr` in normal format.
  - `ls /usr -al`: List all files (including hidden files) under directory `/usr` in long format.
- `man`: Show manual page of Linux. If you want to check the specific usage of a command, tool, or function, you may use this command.
  - `man ls`
  - `man pthread_create`

Basic commands in Linux terminal

- `mkdir dir`: Create a new directory.
- `rmdir dir`: Remove an empty directory (Must be empty).
- `rm`: Removal command.
  - `rm file`: Remove a regular file.
  - `rm -r dir`: Recursively remove a directory file.
  - `-i`: Prompt before every removal.
  - `-f`: Never prompt even if the file is not exist.
- `touch file`: Create a new regular file.
- `cp`: Copy command.
  - `cp file1 file2`: Copy content of `file1` into `file2`.
  - `cp file dir`: Copy `file` into directory `dir`.
  - `cp -r dir1 dir2`: Recursively copy directory `dir1` into directory `dir2`.

Basic commands in Linux terminal

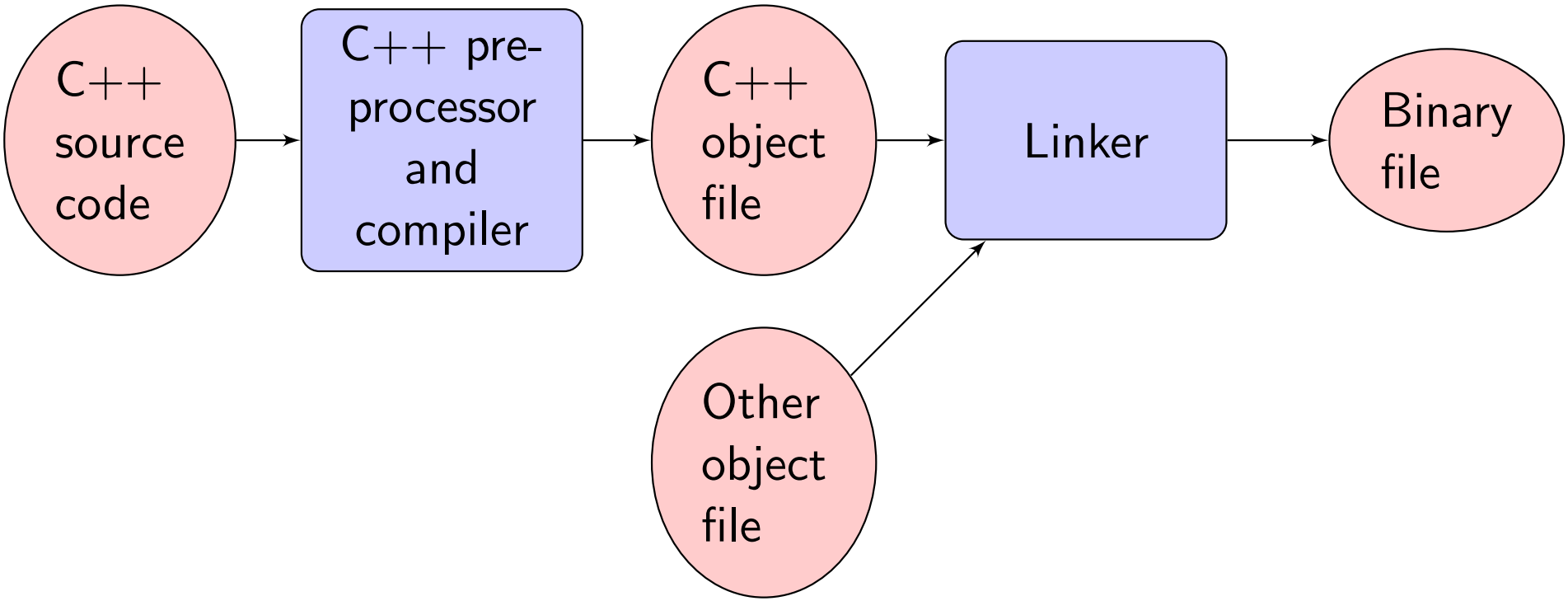
- `mv`: Move command.
  - `mv file1 file2`: Rename `file1` to `file2`.
  - `mv file dir`: Move `file` into directory `dir`.
  - `mv dir1 dir2`: Move directory `dir1` into `dir2`. If `dir2` doesn't exist, then rename `dir1` to `dir2`
- Edit or show a file: `less`, `vi`, `gedit`
- `cat`: Output file content through standard output.
- IO redirection:
  - <: redirect standard input from a file, e.g. `./print < file.in`
  - >: redirect standard output to a file, e.g. `./print < file.in > file.out`
- `diff file1 file2`: File compare.

Basic commands in Linux terminal

- sudo: Short for "superuser do", meaning to execute an executable file with superuser right.
- sudo apt-get install g++: Install g++ in Linux.
- sudo apt-get autoremove g++: Remove g++.

Developing program on Linux

- The flow chart of compiling a C++ program is shown below:



Developing program on Linux

- Compiler: `g++`
- Compiling command: `g++ -o prog prog.cpp lib.cpp`
- `-o` specifies the name of the executable which is the name followed by it.
- You need to put all source files you need (`.cpp`) afterwards, and you don't need to add header files (`.h`).
- The whole process can be regarded as two sub-procedures:
  - Compile: `g++ -c prog.cpp`, `g++ -c lib.cpp`
  - Link: `g++ -o prog prog.o lib.o`
- Other flags:
  - `-g`: Put executable file in the executable file.
  - `-Wall`: Turn on warnings.

Developing program on Linux

- Two types of files: header file and C++ source file.
- Header file: `.h` file includes only class definitions and function declarations.
- C++ source file: `.cpp` file includes class implementations and class definitions.

lib.h

```
#ifndef LIB_H
#define LIB_H
int add(int a, int b);
#endif
```

lib.cpp

```
#include lib.h
int add(int a, int b) {
    return a + b;
}
```

Header guard

- The `LIB_H` defined in the `lib.h` is called header guard. If `LIB_H` was defined before, the block within `#ifndef` and `#endif` will not be shown to the compiler.

test.cpp

```
#include "lib1.h"
#include "lib2.h
int main() {
    Lib0Class lib0(0);
    return 0;
}
```

lib0.h

```
#ifndef LIB0_H
#define LIB0_H
class Lib0Class {
    int a;
public:
    Lib0Class (int x);
};
#endif
```

Header guard

lib1.h

```
#ifndef LIB1_H
#define LIB1_H
#include "lib0.h"
#endif
```

lib2.h

```
#ifndef LIB2_H
#define LIB2_H
#include "lib0.h"
#endif
```

Makefile

- Makefile is a script which defines a rule for code compilation.
- To use the Makefile, you only need to type: make in the command window.

Makefile example

```
all: run_add
run_add: run_add.o add.o
    g++ -o run_add run_add.o add.o
run_add.o: run_add.cpp
    g++ -c run_add.cpp
add.o: add.cpp
    g++ -c add.cpp
clean:
    rm -f run_add *.o
```

- `Target: Dependency`
- "Tab" key before each command is necessary.
- When the dependency is more recent than the target, the rule will be executed.

References


Wikipedia
https://en.wikipedia.org/wiki/Unix