Basic Linux commands and Compilation

ITSC 3181 Introduction to Computer Architecture

https://passlab.github.io/ITSC3181/

Department of Computer Science

Yonghong Yan

yyan7@uncc.edu

https://passlab.github.io/yanyh/

Contents

- Basic Linux commands
  - We will use Ubuntu Linux VM on the lab machine
  - You can use the VM on your own computer and laptop as well.
- Compiling and linking

Linux Basic Commands

It is all about dealing with files and folders
Linux folder: /home/yanyh/…

- ls (list files in the current folder)
  - $ ls -l
  - $ ls -a
  - $ ls -la
  - $ ls -l --sort=time
  - $ ls -l --sort=size –r
- cd (change directory to)
  - $ cd /usr/bin
- pwd (show current folder name)
  - $ pwd
- ~ (home folder)
  - $ cd ~
- ~*user (home folder of a user)*
  - $ cd ~weesan

- What will "cd ~/weesan" do?

- rm (remove a filer/folder)
  - $ rm foo
  - $ rm -rf foo
  - $ rm -i foo
  - $ rm -- -foo
- cat (print the file contents to terminal)
  - $ cat /etc/motd
  - $ cat /proc/cpuinfo
- cp (create a copy of a file/folder)
  - $ cp foo bar
  - $ cp -a foo bar
- mv (move a file/folder to another location. Used also for renaming)
  - $ mv foo bar
- mkdir (create a folder)
  - $ mkdir foo

Basic Commands (cont)

- df (Disk usage)
  - $ df -h /
  - $ du -sxh ~/
- man (manual)
  - $ man ls
  - $ man 2 mkdir
  - $ man man
  - $ man -k mkdir
- Manpage sections
  - 1 User-level cmds and apps
    - /bin/mkdir
  - 2 System calls
    - int mkdir(const char *, …);
  - 3 Library calls
    - int printf(const char *, …);

Search a command or a file

- which
  - $ which ls
- whereis
  - $ whereis ls
- locate
  - $ locate stdio.h
  - $ locate iostream
- find
  - $ find / | grep stdio.h
  - $ find /usr/include | grep stdio.h

Smarty

1. **[Tab] key:** auto-complete the command sequence
2. ↑ **key:** to find previous command
3. **[Ctl]+r key**: to search previous command

Editing a File: Vim

- 2 modes
  - Input mode
    - ESC to back to cmd mode
  - Command mode
    - Cursor movement
      - h (left), j (down), k (up), l (right)
      - ^f (page down)
      - ^b (page up)
      - ^ (first char.)
      - $ (last char.)
      - G (bottom page)
      - :1 (goto first line)
    - Swtch to input mode
      - a (append)
      - i (insert)
      - o (insert line after
      - O (insert line before)
- Delete
  - dd (delete a line)
  - d10d (delete 10 lines)
  - d$ (delete till end of line)
  - dG (delete till end of file)
  - x (current char.)
- Paste
  - p (paste after)
  - P (paste before)
- Undo
  - u
- Search
  - /
- Save/Quit
  - :w (write)
  - :q (quit)
  - :wq (write and quit)
  - :q! (give up changes)

C Hello World

- vi hello.c
- Switch to editing mode: i or a
- Switching to control mode: ESC
- **Save a file: in control mode, :w**
- **To quit, in control mode, :q**
- To quit without saving, :q!

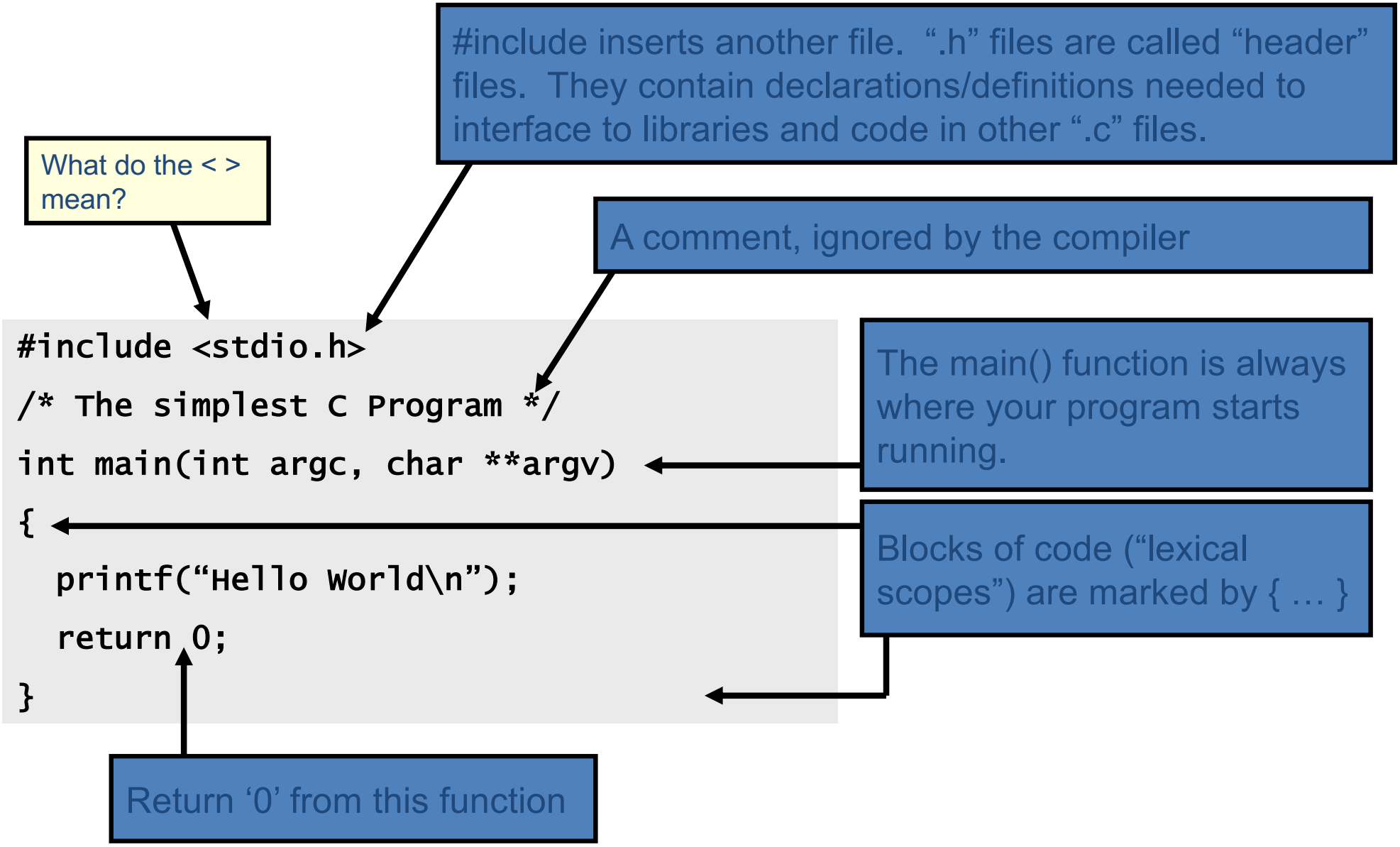
```
#include <stdio.h>
/* The simplest C Program
int main(int argc, char **ar
  printf("Hello World\n");
  return 0;
}
```

- Copy/paste a line: yy and then p, both from the current cursor
  - 5 line: 5yy and then p
- To delete a whole line, in control mode, : dd

Other Editors to Use

- Sublime, Emacs, etc

C Syntax and Hello World



Compilation Process in C

- Compilation process: gcc hello.c -o hello
  - Constructing an executable image for an application
  - FOUR stages
  - Command:
    gcc <options> <source_file.c>

- Compiler Tool
  - gcc (GNU Compiler)
    - man gcc (on Linux m/c)
  - icc (Intel C compiler)

4 Stages of Compilation Process

Preprocessing
gcc -E hello.c -o hello.i
hello.c → hello.i

Compilation (after preprocessing)
gcc -S hello.i -o hello.s

Assembling (after compilation)
gcc -c hello.s -o hello.o

Linking object files
gcc hello.o -o hello

Output → Executable (a.out)
Run → ./hello (Loader)

4 Stages of Compilation Process

1. Preprocessing (Those with # …)
   - Expansion of Header files (#include …)
   - Substitute macros and inline functions (#define …)
2. Compilation
   - Generates assembly language
   - Verification of functions usage using prototypes
   - Header files: Prototypes declaration
3. Assembling
   - Generates re-locatable object file (contains m/c instructions)
   - nm app.o
     0000000000000000 T main
                      U puts
   - nm or objdump tool used to view object files

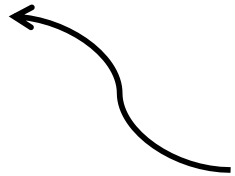
4 Stages of Compilation Process (contd..)

4. Linking
   - Generates executable file (nm tool used to view exe file)
   - Binds appropriate libraries
     - Static Linking
     - Dynamic Linking (default)

- Loading and Execution (of an executable file)
  - Evaluate size of code and data segment
  - Allocates address space in the user mode and transfers them into memory
  - Load dependent libraries needed by program and links them
  - Invokes Process Manager → Program registration

Compiling a C Program

- *gcc <options> program_name.c*



Four stages into one

- Options:

  -Wall: Shows all warnings
  -o output_file_name: By default a.out executable file is created when we compile our program with gcc. Instead, we can specify the output file name using "**-o**" option.
  -g: Include debugging information in the binary.

- man gcc