Accessing ilab, writing, compiling, and running your first C program

Yipeng Huang

Rutgers University

January 23, 2023

Table of contents

Announcements

Accessing materials

- Class canvas: `https://rutgers.instructure.com/courses/210629`
- Most of the time, materials are on the Modules page
- Long range syllabus linked from Canvas:
  `https://yipenghuang.com/teaching/2023-spring/`

Recitations

- Recitations start today.
- More information here:
  `https://rutgers.instructure.com/courses/210629/pages/`
  `recitation-and-office-hour-information?module_item_id=`
  `7956869`
- Review the material and get help on the programming assignments.

Table of contents

Why use Linux?

Do you have Linux? Trick question...

Why use Linux?

- ▶ Stable
- ▶ Open source
- ▶ Flexible: all form factors (wearables, IoT, Raspberry Pi, Roku, Android, laptops, iLab, web hosting, warehouse-scale datacenters)
- ▶ A critical piece of infrastructure for practicing computer science

Key steps to get going

1. Activate account: `https://services.cs.rutgers.edu/accounts/`
2. Familiarize yourself with CS department infrastructure: `https://resources.cs.rutgers.edu/docs/new-users/beginners-info/`
3. Use what you are familiar with to log onto iLab remotely. Command line: Windows command line, macOS, terminal, PuTTy. Graphical: X2Go... `https://resources.cs.rutgers.edu/docs/other/working-at-home/`
4. Use what you are familiar with to move files. SCP, Filezilla, Cyberduck... `https://resources.cs.rutgers.edu/docs/file-storage/accessing-files-remotely/`
5. Use what you are familiar with to edit files. Vim, Emacs, other text editors, VS Code...

Table of contents

Accessing the programming assignments

1. Canvas: `https://rutgers.instructure.com/courses/210629/assignments/2449056?module_item_id=7956894`
2. GitHub: `https://github.com/yipenghuang0302/2023_0s_221`
3. Cloning a repository
4. git: version control tool; interfaces with github.com

Preview of the plan for programming assignments

1. PA0, 1.25% of course grade, five day work time
2. PA1, 6.25% of course grade, two week work time
3. PA2, 12.5% of course grade, two week work time
4. PA3, 12.5% of course grade, two week work time
5. PA4, 12.5% of course grade, three week work time
6. PA5, 12.5% of course grade, two week work time
7. PA6, 12.5% of course grade, two week work time

Why study C?

C is fast
- C is close to the machine: directly manipulate memory
- C is compiled
- C is strongly typed

C is influential
- C is old
- C has a relatively short specification
- Influenced Java, Python, C++, and most major programming languages
- https://www.tiobe.com/tiobe-index/

See one, do one, teach one

To practice programming, you can...

- ▶ Read about it from a book (e.g., Modern C).
- ▶ Listen about it from someone talking about concepts (e.g., lecture slides in Supplementary Reading).
- ▶ Watch someone else program.
- ▶ Practice it.
- ▶ Explain / teach it to a friend.

Table of contents

`rootFinder`: A program that prints square roots if integer

- Headers
- Command line arguments
- Opening files
- Reading from files
- `printf` and format specifiers
- `EXIT_SUCCESS`

Including headers

- ► `#include <stdio.h>`
- ► `#include <stdlib.h>`
- ► `#include <stdbool.h>`
- ► `#include <math.h>`

Command line arguments: First encounter with pointers
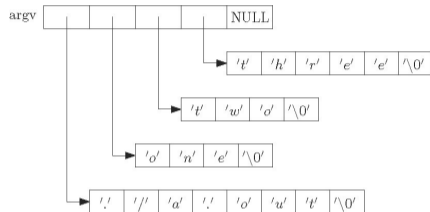
What is `char* argv[]`

Figure: Image credit: `http://www.csc.villanova.edu/~mdamian`

In C, Strings, `char*`, and `char[]` are all the same

▶ `char greeting[6] = {'H','e','l','l','o','\0'};`

▶ `char greeting[] = "Hello";`

Opening a file

The mode in `FILE *fopen(const char *filename, const char *mode)`

- ► `"r"`: read from the file
- ► `"w"`: write, starting at the beginning of the file
- ► `"a"`: write, starting at the end of the file (append)

Reading characters from a file

- `int fgetc(FILE *stream)`
- `char *fgets(char *str, int n, FILE *stream)`
- `int fscanf(FILE *stream, const char *format, ...)`

Printing to command line

The format string in `printf(char* format, args)`

- `%d`: integer
- `%ld`: long integer
- `%s`: string
- `%c`: character
- `%f`: float

Compiling and running your program

How does a program end up on your computer?

```
gcc -Wall -Werror -fsanitize=address -std=c99 -o
rootFinder rootFinder.c -lm
```

- ▶ `gcc`: GNU C Compiler
- ▶ `-Wall -Werror`: Enable helpful warnings.
- ▶ `-fsanitize=address`: Enable memory checking.
- ▶ `-std=c99`: Set C standard version number.
- ▶ `-o rootFinder`: Output binary.
- ▶ `rootFinder.c`: Source file.
- ▶ `-lm`: Link the math library implementation.

Compiling and running your program

How does a program end up on your computer?

How a Makefile works

- ▶ `$@`: target file name
- ▶ `$<`: first prerequisite
- ▶ `$^`: all prerequisites