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

In-person lecture for Thursday, September 4 is canceled.
Please watch the pre-recorded session here:
https://www.youtube.com/watch?v=SPzHF3YqUNw

Yipeng Huang

Rutgers University

September 4, 2025

Table of contents

Announcements

Accessing materials

- Class canvas: `https://rutgers.instructure.com/courses/355196`
- Most of the time, materials are on the Modules page
- Long range syllabus linked from Canvas: `https://yipenghuang.com/teaching/2025-fall-211/`

Recitations

- Recitations start Tuesday, 9/9.
- More information here: `https://rutgers.instructure.com/courses/355196/pages/recitation-and-office-hour-information?module_item_id=12376605`
- 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/355196/assignments/3779232?module_item_id=12376634`
2. GitHub: `https://github.com/yipenghuang0302/2025_1f_211`
3. Cloning a repository
4. git: version control tool; interfaces with github.com

Preview of the plan for programming assignments

1. PA0, 1% of course grade, 10 day work time
2. PA1, 4% of course grade, two week work time
3. PA2, 10% of course grade, two week work time
4. PA3, 10% of course grade, two week work time
5. PA4, 10% of course grade, two week work time
6. PA5, 10% of course grade, two and half week work time
7. PA6, 10% 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[]`

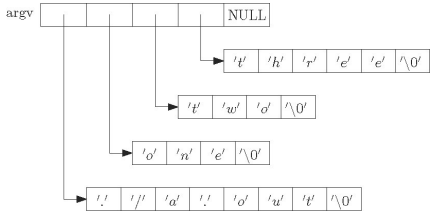

Figure: Image credit: `http://www.csc.villanova.edu/~mdamian`

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, ...)`

Control flow

- Conditionals
- Loops
- `break;`
- `continue;`

Printing to command line

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

Format specifiers we care about now:

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

Comprehensive documentation:
`https://cplusplus.com/reference/cstdio/printf/`

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

Assignment infrastructure for this course

Navigating the 2025_1f_211/ assignments directory

- `autograder.py`
- `tests/`: test cases
- `answers/`: expected answers
- Every assignment part has several fixed test cases for development, several randomized test cases for validation.
- `assignment_autograder.py`
- `tar cvf pa0.tar .`