C Programming: I/O, files

Yipeng Huang

Rutgers University

January 26, 2023

Table of contents

Announcements
    Class resources

rootFinder: A program that prints square roots if integer
    Program structure
    Opening a file
    Reading characters from a file
    Control flow
    Printing to command line
    Compiling and running your program

pointers.c: A lab exercise for pointers, arrays, and memory
    Lesson 1: What are pointers?
    Lesson 2: Dereferencing pointers with *
    Lesson 3: The integer datatype uses four bytes

Class resources

- You should notice now these slides are not comprehensive.
- Supplemental reading and recitations slides on Canvas.
- Sequence of recitations this afternoon.
- Programming assignment 0 progress?
- Where have you found help?
- Piazza.

Table of contents

Announcements

- Class resources

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

- Program structure
- Opening a file
- Reading characters from a file
- Control flow
- Printing to command line
- Compiling and running your program

`pointers.c`: A lab exercise for pointers, arrays, and memory

- Lesson 1: What are pointers?
- Lesson 2: Dereferencing pointers with *
- Lesson 3: The integer datatype uses four bytes

`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[]`

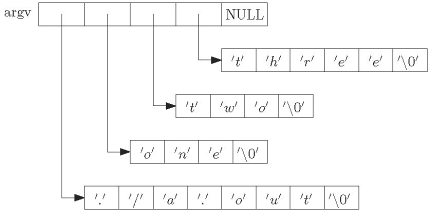

Figure: Image credit: `http://www.csc.villanova.edu/~mdamian`

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
- `for` loops
- `while` loops
- `do-while` 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

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

Assignment infrastructure for this course

Navigating the 2023_0s_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 validataion.
- `assignment_autograder.py`
- `tar cvf pa0.tar .`

Table of contents

Announcements
- Class resources

rootFinder: A program that prints square roots if integer
- Program structure
- Opening a file
- Reading characters from a file
- Control flow
- Printing to command line
- Compiling and running your program

pointers.c: A lab exercise for pointers, arrays, and memory
- Lesson 1: What are pointers?
- Lesson 2: Dereferencing pointers with *
- Lesson 3: The integer datatype uses four bytes

Lesson 1: What are pointers?

- Pointers are numbers
- The unary operator & gives the “address of a variable”.
- how big is a pointer? 32-bit or 64-bit machine?
- Pointers are typed

Lesson 2: Dereferencing pointers with *

`*pointer`: dereferencing operator: variable in that address

`int* ptr` and `int *ptr`

No difference between `int* ptr` and `int *ptr`

- `int* ptr` emphasizes that `ptr` is `int*` type
- `int *ptr` emphasizes that when you dereference `ptr`, you get a variable of type `int`

Lesson 3: The integer datatype uses four bytes

- Memory is an array of addressable bytes
- Variables are simply names for contiguous sequences of bytes