CS24 Week 1 Lecture 1

Kyle Dewey

Overview

- Why this course?
- Syllabus
- C Review
  - Arrays
  - Multiple files
  - File I/O
  - Command-line arguments

About Me

- 3rd year Ph.D. Candidate, doing programming languages research
- Not a professor
- Just call me Kyle
- Second time teaching, first time teaching CS24
- TA for operating systems, capstone, and programming languages

About this Class

- See something wrong? Want something improved? Email me about it! (kyledewey@cs.ucsb.edu)
- I generally operate based on feedback

Bad Feedback

- This guy sucks.
- This class is boring.
- This material is useless.

Good Feedback

- This guy sucks, *I can't read his writing.*
- This class is boring, *it's way too slow.*
- This material is useless, *I don't see how it relates to anything in reality.*
- I can't fix anything if I don't know what's wrong

Questions

- Which best describes you?
  - Last CS course I will ever take
  - Trying to enter the major
  - In the major already
  - Other

Office Hours

Why this Course?

- We have a series of names and addresses, and we want to search for an address given a name
- How do we solve this in C?

Why this Course?

- We have a series of names and addresses, and we want to search for an address given a name
- How do we solve this in C?
- Key question: how many names do we have?
  - Small office? Phone book? Facebook?

Simple

```
int numIndex(int* array,
             int length, int num) {
   int x;
   for(x = 0; x < length; x++) {
      if (array[x] == num)
         return x;
   }
   return -1;
}
```

Sophisticated

```
int binary_search(int* A, int key,
                  int imin, int imax){
  while (imax >= imin){
    int imid = midpoint(imin, imax);
    if(A[imid] == key) {
      return imid;
    } else if (A[imid] < key) {
      imin = imid + 1;
    } else {
      imax = imid - 1;
    }
  }
  return -1;
}
```

http://en.wikipedia.org/wiki/Binary_search_algorithm

Why this Course?

- Data representation is vital to **any** problem
  - Large data usually demands sophisticated data representations
  - Small data typically demands simpler approaches

Why this Course?

- A discussion of more sophisticated techniques, along with:
  - When to use them
  - How to implement them

Syllabus

C Review

Arrays

- What is an array?
- What is a string?

Arrays

- How do I make an array of integers?
  - With initial contents?

Arrays

- How do I access an element of an array?
- How do I update an element of an array?

Arrays - Pitfalls

```
int arr[3] = {0, 1, 2};
int bad1 = arr[-1];
int bad2 = arr[3];
```

Arrays - Pitfalls

```
int arr[3];
int bad3 = arr[0];
```

Undefined Behavior

- C (and C++) standard says you won't do that
  - You did that
  - Why did you do that?
- Means your program's meaning is undefined - all interpretations are valid!

Put it to the Test

- Write a function named `findmax`, which takes:
  - An array of integers
  - The length of the array of integers
- It returns the maximum integer in the array
- Pause - any unanswered questions?

Put it to the Test

- Write a function named `findmax`, which takes:
  - An array of integers
  - The length of the array of integers
- It returns the maximum integer in the array**, or -1 for an empty array**

Multiple Files

Situation 1

- You have written a *library* of routines for manipulating images
- You want to share these with other programmers
- How can we go about this?

Situation 2

- You are at Google working on their search engine
- The search engine is divided into these components:
  - An external interface
  - A database of various webpages
  - A sophisticated search algorithm
- How can all parties work together?

Situation 3

- You are working on a large project, and putting everything in one file leads to a mess
  - 10s of thousands of lines of code
  - By the time you're at line 2,000, you can't remember what 200 did
  - Editing is a nightmare

Solution: Multiple Files

- Splitting code up into multiple files allows for easier collaboration, and helps *hide details* from us
- Generally, the fewer details you must know, the better
  - Mark of good software design

Header Files

- In C/C++, this is accomplished via header files
- A header file defines an *interface*
- Code can *include* other header files to gain access to the interfaces
- The interfaces are implemented in separate files

Header Files Example

Basic File I/O

Question

- Say a program is not permitted to read or write to files, the terminal, the network, or any other source
- Can the resulting program do anything useful?

I/O (Input/Output)

- The way programs interact with the outside world
- Without it, programs are simply things that turn computers into space heaters

File I/O

- When working with files, we must `open` a file before we can `read` from it
- When we are done with a file, we must `close` it
  - What happens if we forget to close it?

Reading from a File

- Can read one character at a time
  - See `cat1.c`, which uses `fgetc` for this

Reading from a File

- Can also read multiple characters at a time
  - See `cat2.c`, which uses `fgets` for this

Questions

- What extra bit is needed to read multiple characters at a time?
    - What happen if we get this extra bit wrong?
- Why read multiple characters at a time?

Command Line Arguments

UNIX Commands

- We have seen a bunch of UNIX commands used at this point
- How exactly do these programs interpret what they are supposed to do?
  - How does `emacs` know which file to open?
  - How does `cd` know which directory to go to?

Command Line Arguments

- A standard way to tell programs what and how to do
- In C/C++, we can get access to the command line arguments via the parameters to the `main` function

Command Line Arguments Example (`echo.c`)

Command Line Arguments

- What is `argc`? What is it set to?
- What is `argv`? What is it set to?