1x-x13 Recitation: C Review

Monday, February 21st, 2022

Agenda

- Logistics
- C Review
- Activity 1: Getopt
- Activity 2: Pythagorean Solver

Logistics

- Cache Lab is due next Thursday, March 3rd
  - Come to office hours for help

C Review

C Review: Pointers

- Pointer: stores address of some value in memory
- Dereferencing a NULL pointer causes segfault

- Dereferencing a pointer: *p
- Access address of a value: p = &v

C Review: Pointers

- What is wrong with this code?

```
int main(int argc, char** argv) {
    int *a = (int*) malloc(213 * sizeof(int));
    for (int i=0; i<213; i++) {
        if (a[i] == 0) a[i]=i;
        else a[i]=-i;
    }
    return 0;
}
```

C Review: Pointers

- `malloc` can fail!

```
int main(int argc, char** argv) {
    int *a = (int*) malloc(213 * sizeof(int));
    if (a == NULL) return 0;
    for (int i=0; i<213; i++) {
        if (a[i] == 0) a[i]=i;
        else a[i]=-i;
    }
    return 0;
}
```

C Review: Pointers

- Allocated memory is not initialized!

```c
int main(int argc, char** argv) {
    int *a = (int*) calloc(213, sizeof(int));
    if (a == NULL) return 0;
    for (int i=0; i<213; i++) {
        if (a[i] == 0) a[i]=i;
        else a[i]=-i;
    }
    return 0;
}
```

C Review: Pointers

- All allocated memory must be freed!

```c
int main(int argc, char** argv) {
    int *a = (int*) calloc(213, sizeof(int));
    if (a == NULL) return 0;
    for (int i=0; i<213; i++) {
        if (a[i] == 0) a[i]=i;
        else a[i]=-i;
    }
    free(a);
    return 0;
}
```

C Review: Arrays

- Initializing your array
  - `int *a = calloc(4, sizeof(int));`
    - `Allocated on Heap`
  - `int a[4];`
    - `Allocated on stack`

- Where does the following point to?

```
int a[4] = {1,2,3,4};
```
- `a[0]`
- `*(a + 3)`

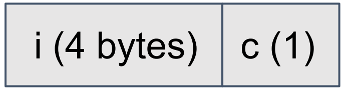
```
char *listOfName[4] = {"Alice", "Bob", "Cherry"};
```
- `(listofName + 1)`
- `*(listOfName + 1)`

C Review: Structs + Unions

Struct:

- Groups list of variables under one block in memory

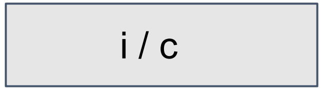
```
struct temp {
     int i;
     char c;
};
```



Union:

- Store different data types in same region of memory
- Many ways to refer to same memory location

```
union temp {
     int i;
     char c;
};
```



C Review: Valgrind

- What is Valgrind?
  - Tool used for debugging memory use
- Valgrind may…
  - Find corrupted memory
  - Find potential memory leaks and double frees
  - Detects invalid memory reads and writes
- To learn more… man valgrind

Valgrind Demo

- Even if program seems to run successfully, Valgrind can uncover memory leaks and invalid writes

C Review Conclusion

- Did you know each concept? If not…
  - Refer to the C Bootcamp slides

- Were the concepts so easy you were bored? If not…
  - Refer to the C Bootcamp slides

- When in doubt…
  - Refer to the C Bootcamp slides

- This will be *very* important for the rest of this class, so make sure you are comfortable with the material covered or come to the C Bootcamp!

C Programming Style

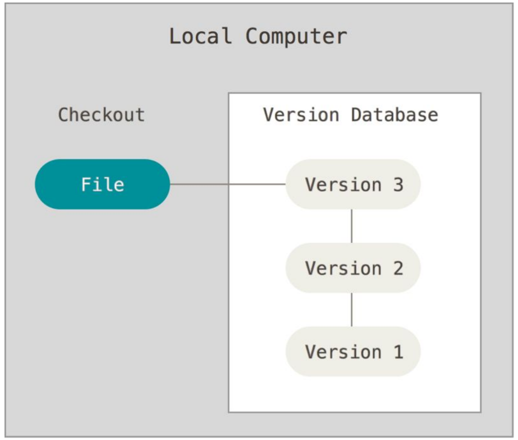
- Write comments and then implement functionality
- Communicate meaning through naming choices
- Code should be testable. Modularity supports this
- Use consistent formatting
- Common bugs: memory and file descriptor leaks, check errors and failure conditions

Introduction to Git

Version control is your friend

What is Git?

- Most widely used version control system out there
- Version control:
    - Help track changes to your source code over time
    - Help teams manage changes on shared code



Git Commands

- Clone: git clone <clone-repository-url>
- Add: git add . or git add <file-name>
- Push / Pull: git push / git pull
- Commit: git commit -m “your-commit-message”
    - Good commit messages are key!
    - Bad:“commit”, “change”, “fixed”
    - Good: “Fixed buffer overflow potential in AttackLab”

Activity 0 + 1

```
$ wget http://www.cs.cmu.edu/~213/activities/rec6.tar
$ tar xvpf rec6.tar
$ cd rec6
$ make
```

Activity 0: reading `man` pages!

- Reading `man` pages is important!
- To get started, either:
  - `$ man 3 getopt` on Terminal
  - Google “man getopt”

- Overall, what does getopt do?
- What arguments does it take?
- How can you use it in a program?
- https://linux.die.net/man/3/getopt

Activity 2

Let’s write a Pythagorean Triples Solver!

- Open `pyth_solver.c` in a text editor of your choice.

- Your code should:
  - Take in args with `a`, `b`, `c` flags
  - Determine if the a,b,c is a Pythagorean triple
  - Error check on: number and validity of args (exit on invalid args)
  - Invalid: too few or negative args
  - Verbose mode: output `a^2`, `b^2`, `c^2`

C Hints and Math Reminders

- Can your Pythagorean Triple parse these input?
  - 3, 4, 5
  - 5, 12, 13
  - 7, 24, 25

- $a^2 + b^2 = c^2$
  - $\Rightarrow a = \sqrt{c^2 - b^2}$
  - $\Rightarrow b = \sqrt{c^2 - a^2}$
  - $\Rightarrow c = \sqrt{a^2 + b^2}$
  - $\Rightarrow 3^2 + 4^2 = 5^2$

- String to float in C:

```
#include <stdlib.h>
float atof(const char *str);
```

- Square root in C:

```
#include <math.h>
float sqrt(float x);
```

How to compile and run your solver

```
$ gcc pyth_solver.c
$ ./a.out (ARGS)
```

More details on handout!

Good luck!

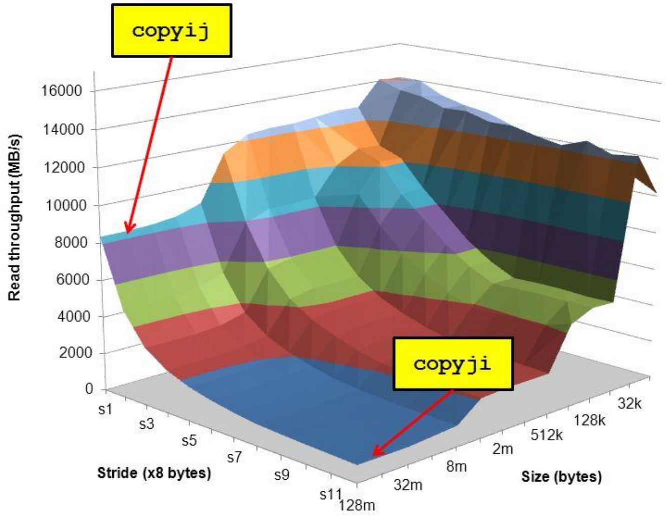
Looking Ahead

Cache Lab Overview

- Programs exhibiting locality run *a lot* faster!
  - Temporal Locality – same item referenced again
  - Spatial Locality – nearby items referenced again

- Cache Lab's Goal:
  - Understand how L1, L2, … etc. caches work
  - Optimize memory dependent code to minimize cache misses and evictions
    - Noticeable increase in speed

- The use of git is required
  - Commit regularly with meaningful commit messages



If you get stuck…

- Reread the writeup
- Look at CS:APP Chapter 6
- Review lecture notes (http://cs.cmu.edu/~213)
- Come to Office Hours (See piazza post on OH for times & locations)
- Post private question on Piazza
- `man malloc`, `man valgrind`, `man gdb`

Cache Lab Tips!

- Review cache and memory lectures
  - Ask if you don’t understand something
- Start early, this can be a challenging lab!
- Don’t get discouraged!
  - If you try something that doesn't work, take a well deserved break, and then try again
- Good luck!

Appendix

Appendix: `Valgrind`

- Finding memory leaks
  - `$ valgrind –leak-resolution=high –leak-check=full –show-reachable=yes –track-fds=yes ./myProgram arg1 arg`
- Remember that Valgrind can be used for other things, like finding invalid reads and writes!

Appendix: `$ man 3 getopt`

- `int getopt(int argc, char * const argv[], const char *optstring);`
  - `int argc` → argument count passed to `main()`
    - Note: includes executable, so `./a.out 1 2` has `argc=3`
  - `char * const argv` is argument string array passed to `main`
  - `const char *optstring` → string with command line arguments
    - Characters followed by colon require arguments
      - Find argument text in `char *optarg`
    - `getopt` can't find argument or finds illegal argument sets `optarg` to "?"
    - Example: "`abc:d:`"
      - `-a -b -c 3 -d 4`
      - a and b are boolean arguments (not followed by text)
      - c and d are followed by text (found in `char *optarg`)
- Returns: `getopt` returns -1 when done parsing

Appendix: Clang / LLVM

- Clang is a (gcc equivalent) C compiler
  - Support for code analyses and transformation
  - Compiler will check your variable usage and declarations
  - Compiler will create code recording all memory accesses to a file
  - Useful for Cache Lab Part B (Matrix Transpose)