15-213 Recitation: How to Succeed in 213

January 23, 2023

Agenda

- Introduction
- Course Details
- Office Hours
- 213 Advice from TAs
- Data Lab
- Looking ahead to Bomblab

Introduction

- Welcome to 15-213/14-513/15-513!
- Recitations are for…
  - Reviewing lectures
  - Discussing homework problems
  - Interactively exploring concepts
  - Previewing future lecture material

- Please, **please** ask questions!

Course Details

- How do I get help?
  - Course website: http://cs.cmu.edu/~213
  - Office hours
  - Piazza
  - *Definitely* consult the course textbook
  - **Carefully read the assignment writeups!**
- All labs are submitted on Autolab.
- All labs should be worked on using the **shark machines.**
- Can download labs 0-3 using:
  autolab download 15213-s23:<lab_name>
  cd into the datalab folder; tar -xf <lab_name>.tar

Office Hours

- Queue link: https://ohq.eberly.cmu.edu/#/courses
- Please locate the TA in the specified location!
- Current OH schedule
  - Mon: 6-10pm
  - Tues: 6-10pm
  - Wed: 6-10pm
  - Thurs: 6-10pm
  - Fri: 4-8pm
  - Sat & Sun: 2-6pm (Remote)

OH Etiquette

- Office hours are for getting ideas on how to debug or better approach your homework!
- Please try to narrow down your problem area as much as possible to help TAs help you!
- **Write a description!** If you don’t have a description, you may be frozen/removed from the queue. Make sure to use the tags!
- TAs will only spend 10 minutes per student and then you can rejoin the queue.
- We will close the queue early so everyone can be helped so please keep this in mind!

How to Succeed at 213

Some advice from your friendly TAs ;)

What is success in 213?

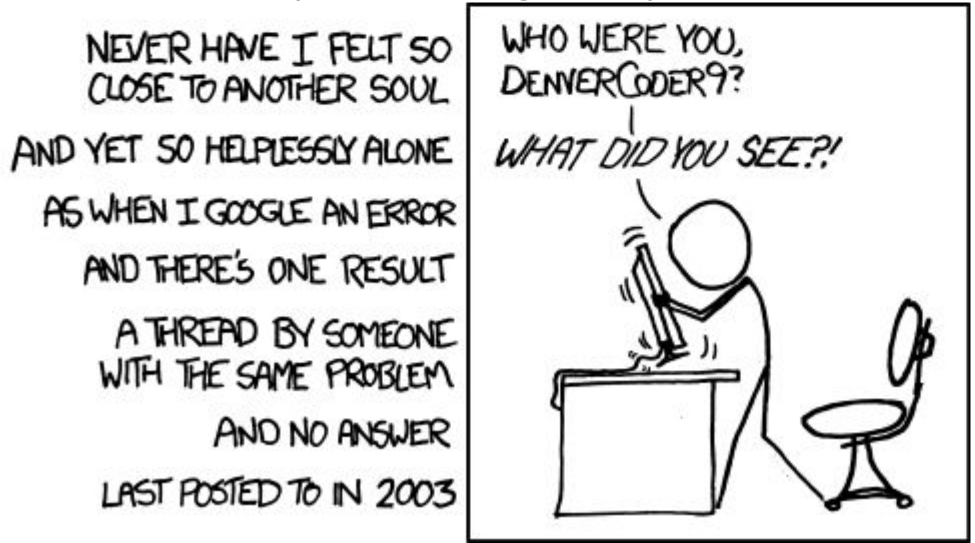
- Some of you (probably most) see success as an A
- … buuuutttttt you can still succeed without getting an A, in fact, true success in 213 is **learning the material**
- And this can be difficult because we will cover *a lot* of different topics, many of which will probably be new to you (and that's okay!)

How do I learn the material then?

- Engage with the topics in lecture
- Read the textbook
- Don’t wait to learn the material when you need to use it
- Ask questions!

I've tried that, but I'm confused. Now what?

- It's okay to be confused! These topics can be difficult and take time to truly understand
- (Some) online resources are okay to use, but a general google search probably won't give you helpful results . . .


NEVER HAVE I FELT SO CLOSE TO ANOTHER SOUL
AND YET SO HELPLESSLY ALONE
AS WHEN I GOOGLE AN ERROR
AND THERE'S ONE RESULT
A THREAD BY SOMEONE WITH THE SAME PROBLEM
AND NO ANSWER
LAST POSTED TO IN 2003
WHO WERE YOU, DENVERCODER9?
WHAT DID YOU SEE?!

https://xkcd.com/979/

I need help with a concept

- Read the textbook
- Come to OH and ask your TAs ;)
- Come to Prof. OH (they are approachable!)
- Ask on piazza
- Ask your recitation TAs to cover the topic again

I need help with a problem or bug

- Step away and come back after a small break
- Try to solve on your own (debugging for an hour is not that long)
  - Generally, give yourself a day to mull over the problem (your brain will continue to think about it while you do other tasks!)
- If general bug, try some *reputable* sites to find similar problems (see next slide)
- Come to OH!
- Post on piazza!

Actually good online resources

- https://itsfoss.com/linux-man-page-guide/
- https://man7.org/linux/man-pages/
- https://en.cppreference.com/w/c
  - Make sure to use the C (not C++) version!
- https://www.cs.virginia.edu/~evans/cs216/guides/x86.html
- https://beej.us/guide/
- http://www.stackgrowsdown.com/

Other helpful advice!

- Learn GDB early ***before*** you have to rely on it to debug
- Read the writeups (yes, there can be, *and will be*, relevant material on all 20 pages of a writeup)
- Don’t start labs late
- Save some Grace days for malloc (~40 hours is average)
- You don’t have to pass every test case of every assignment
- Be comfortable with the command line (it's not that scary!)
- Be comfortable with different editors (I’m looking at you VScode 👀)
- If you need help, ask! We are here to help YOU!

Data Lab: Getting Started

- Download the handout from autolab
  - Method 1:
    - `scp <path to datalab.tar> <andrewid>@shark.ics.cs.cmu.edu:<my course directory>`
    - `ssh <andrewid>@shark.ics.cs.cmu.edu`
    - `cd to the datalab.tar file`
    - `tar -xf datalab.tar`
  - Method 2:
    - `autolab download 15213-s23:datalab`
    - `cd into the datalab folder`
    - `tar -xf datalab.tar`

Data Lab: Getting Started

- Upload `bits.c` file to Autolab for submission
  - `make submit`

Data Lab: Running your code

- `dlc`: a modified C compiler
- `btest`: runs your solutions on random values
- `bddcheck`: exhaustively tests your solutions
  - Checks all values, formally verifying the solution
- `driver.pl`: Runs both dlc and bddcheck
  - Exactly matches Autolab’s grading script
  - You will likely only need to submit once
- For more information, **read the writeup**
  - Available under autolab as “**View writeup**”
  - **Read the writeup please!**

Data Lab: Reminders

- Casting between **int** and **long** is ok, in either direction
- Be aware of operations and their types!
  - ! returns an **int** *even if the argument is a long*
- Good idea to append "L" suffix to every integer constant
  - `(1L << 63)` is not the same as `1 << 63`
  - `(!x << 63)` is not the same as `((long)!x) << 63`

Divide and Conquer (Bit Count)

Let’s count how many bits are set in a number. For each challenge, you can use any operator allowed in the integer problems in datalab.

Using 1 operator, we return the number of bits set in a 1-bit number:

```
int bitCount1bit(int x) {return x;}
```

Divide and Conquer (cont.)

How about if there are two bits in the input? (4 ops max)

```
int bitCount2bit(int x) {
    int bit1 = ____________________ & ____________________;
    int bit2 = ____________________ & ____________________;
    return ____________________ + bit1 ;
}
```

Divide and Conquer (cont.)

How about if there are four bits? (8 ops max)

```
int bitCount4bit(int x) {
    int mask = ______________________________;
    int halfSum = ______________________________;
    int mask2 = ______________________________;
    return ___________________ + ___________________ ;
}
```

Divide and Conquer (cont.)

How about if there are eight bits? (12 ops max)

```
int bitCount8bit(int x) {
    int mask = ______________________________;
    int quarterSum = ____________________________;
    int mask2 = ______________________________;
    int halfSum = ___________________________________;
    int mask3 = _____________________________;
    return ____________________ + _____________________ ;
}
```

Questions?

- Remember
  - cprogramminglab is due tomorrow!
    - You really should have started already!
  - Bomb Lab comes out this Thursday
  - Data Lab is due next Thursday
- Read the lab writeup!

Looking Ahead… Bomblab!!

What is Bomb Lab?

- An exercise in reading x86-64 assembly code.
- A chance to practice using GDB (a debugger).
- Why?
  - x86 assembly is low level machine code. Useful for understanding security exploits or tuning performance.

  - GDB can save you days of work in future labs **(Malloc)** and can be helpful long after you finish this class.

Downloading Your Bomb

- Here are some highlights of the write-up:
  - Bombs can only run on the shark machines. They fail if you run them locally or on another CMU server.
  - Each bomb is unique - if you download a second bomb, bad things can happen! Stick to only one bomb.
  - Bombs have six phases which get progressively harder.
  - Make sure to read the writeup for more tips and common mistakes you might make.

Detonating Your Bomb

- Blowing up your bomb automatically notifies Autolab
  - ***Dr. Evil*** deducts 0.5 points each time the bomb explodes.
  - It's very easy to prevent explosions using **break points** in GDB. More information on that soon.
- Inputting the correct string moves you to the next phase.
- Don't tamper with the bomb. Skipping or jumping between phases detonates the bomb.
- You have to solve the phases in order they are given. Finishing a phase also notifies Autolab automatically.

Bomb Hints

- ***Dr. Evil*** may be evil, but he isn't cruel. You may assume that functions do what their name implies
  - i.e. phase_1() is most likely the first phase. printf() is just printf(). __isoc99_sscanf() is just a weird name for sscanf().
  - If there is an explode_bomb() function, it would probably help to set a breakpoint there!
- Use the man pages for library functions.
  - Although you can examine the assembly for snprintf(), we assure you that it's easier to use the man pages ($ man snprintf) than to decipher assembly code for system calls.
- Most cryptic function calls you'll see (e.g. callq … <_exit@plt>) are also calls to C library functions.
  - You can safely ignore the @plt as that refers to dynamic linking.

"That's all Folks!"

F22 Bomblab Slides

https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15213-f22/www/recitations/rec04_slides.pdf