CSCI 331: Introduction to Computer Security

Lecture 2: C Review

Instructor: Dan Barowy

Williams

Topics

Drop/add deadline: Friday, 17th of September

More about grades

Anonymous feedback

C review

(for more review, see lectures page on www)

Quiz

Grades

Purpose:

1. to **reduce your stress level** about grades, and
2. to make feedback **actionable**.

Grading

Final project:	20%
Midterm exam:	20%
Programs/Labs:	30%
Writing assignments:	20%
Attendance and class discussion:	10%

Grading

TRADITIONAL GRADING SYSTEM		STANDARDS-BASED SYSTEM	
A	90-100%	4	Proficient on all standards
B	≥ 80% and < 90%	3	Proficient on most standards
C	≥ 70% and < 80%	2	Proficient on half of the standards
D	≥ 60% and < 70%	1	Proficient on less than half of the standards
F	< 60%	0	Missing

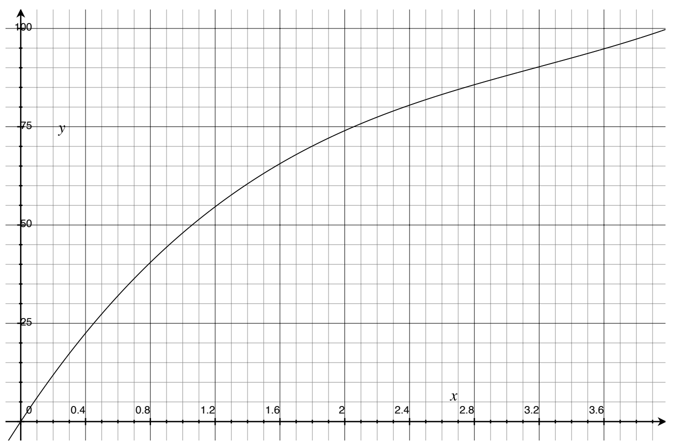
These aren't supposed to line up.

Grading

STANDARDS-BASED SYSTEM		
4	Proficient on all standards	100%
3	Proficient on most standards	88%
2	Proficient on half of the standards	74%
1	Proficient on less than half of the standards	48%
0	Missing	0

Here are the "point conversions"

Grading



Here are the "point conversions"

Grade spreadsheet

Grading

Final project:	20%
Midterm exam:	20%
Programs/Labs:	30%
Writing assignments:	20%
Attendance and class discussion:	10%

No "attendance hacking"!

No more than 3 unexcused absences.

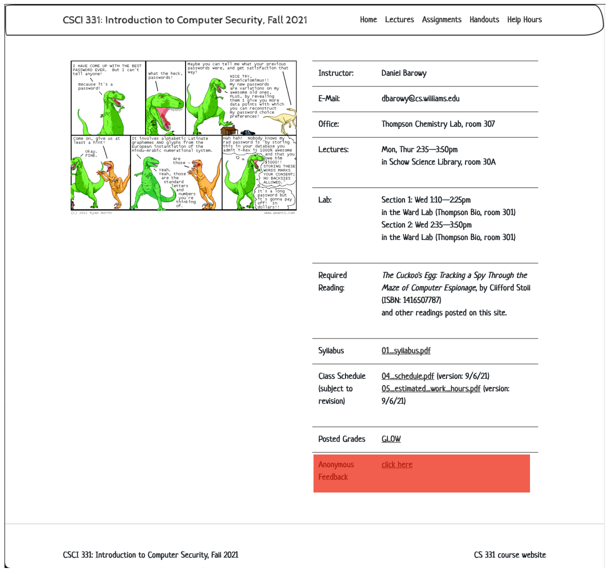
Rarely, I will award bonuses for exceptional work.

Questions?

Feedback

Anonymously or eponymously

Anonymous Feedback


I HAVE COME UP WITH THE BEST PASSWORD EVER. But I can't tell anyone!
Because it's a password!
What the heck, passwords!
Maybe you can tell me what your previous passwords were, and get satisfaction that way!
NICE TRY, Dromiceiomimus!! My new passwords are variations on my awesome old ones, PLUS, by revealing them I give you more data points with which you can reconstruct my password choice preferences!
Come on, give us at least a hint!
Okay. FINE.
It involves alphabetic Latinate graphemes AND glyphs from the European instantiation of the Hindu-Arabic numerational system.
Are those -
Yeah. Yeah, those are the standard letters and numbers you're thinking of.
Hah hah! Nobody knows my rad password is "by storing this in your database you admit T-Rex is 1000% awesome and that you owe him $5000!! STORING THESE WORDS MARKS YOUR CONSENT; NO BACKSIES ALLOWED."
It's a long password but it's gonna pay off! In dollars!!
(C) 2012 Ryan North
www.qwantz.com

CSCI 331: Introduction to Computer Security, Fall 2021 — Home Lectures Assignments Handouts Help Hours

Instructor:	Daniel Barowy
E-Mail:	dbarowy@cs.williams.edu
Office:	Thompson Chemistry Lab, room 307
Lectures:	Mon, Thur 2:35—3:50pm in Schow Science Library, room 30A
Lab:	Section 1: Wed 1:10—2:25pm in the Ward Lab (Thompson Bio, room 301) Section 2: Wed 2:35—3:50pm in the Ward Lab (Thompson Bio, room 301)
Required Reading:	*The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*, by Clifford Stoll (ISBN: 1416507787) and other readings posted on this site.
Syllabus	01_syllabus.pdf
Class Schedule (subject to revision)	04_schedule.pdf (version: 9/6/21) 05_estimated_work_hours.pdf (version: 9/6/21)
Posted Grades	GLOW
Anonymous Feedback	click here

CSCI 331: Introduction to Computer Security, Fall 2021 — CS 331 course website

Your to-dos

1. Answer "Getting to Know You" survey **by tomorrow**.
2. Reading response (Stoll) **due Wed**.
   i. LaTeX optional.
   ii. Must be printed, put in my box.
3. Sign and return Code of Ethics **by Wed**.
   1. Put in my box.
4. First lab **on Wednesday**.
   Do you know what section you are in?

Readings for Lab 0

1. Lab 0 writeup.
   Not a bad idea to skim labs ahead of time.

Lab 0

If you have a **laptop** that you plan to use for the semester, please **bring it** to our **first lab meeting**.

If you prefer to use a **lab machine**, you **don't need to bring anything**.

The C Programming Language

Activity: What do you know about C?

Let's start with the easy stuff

```
$ gcc helloworld.c
```

Like Java, C programs need to be **compiled** before you can run them.

The C compiler ignores many problems

```
$ gcc -Wall helloworld.c
```

So you should always ask it to report **warnings**.

If you don't like a.out

```
$ gcc -Wall helloworld.c -o helloworld
```

Tell the compiler what you want the output **named**.

C Background

1. Despite its quirks, it has many of the **features that you know and love** in Java/Python, etc. (it looks sort of like Java!)
2. Often used in **low-level** or "systems" programming.
3. Nearly as **fast** as expert assembly code; usually faster than non-expert assembly.
4. **No safety net**. Very easy to write programs with subtle bugs.
   1. **No garbage collector: no memory safety.**
   2. **No bounds checker: off-by-one is subtle!**
   3. **No objects: roll your own!!**
   4. **No strings: null-terminated char arrays!!!**
   5. **This list is not exhaustive!!!!**

The problem with C is **not** its **complexity**. The problem is its **simplicity**.

Remember these **rules** and you'll be **OK**!

Rule 0:

Pointers are for **pointing at** other values in **memory**.

```
#include <stdio.h>

int main() {
  int num = 4;
  int *num_ptr = &num;
  printf("num = %d, and it is stored at %p.\n", num, num_ptr);
  return 0;
}
```

Rule 1:

Whenever you **store** a **variable**, you **always** ask C to **reserve memory** for some **duration**.

```c
#include <stdio.h>

int main() {
  int num = 331;
  printf("%d rocks!\n", num);
  return 0;
}
```

short (automatic)

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
  int *num_ptr = malloc(sizeof(int));
  if (!num_ptr) {
    printf("Unable to allocate.\n");
    exit(1);
  }
  *num_ptr = 331;
  printf("%d rocks!\n", *num_ptr);
  return 0;
}
```

long (allocated)

Activity: What **effect** do these programs have on **memory**?

Rule 2:

All long duration storage needs to be both **allocated** and **deallocated**.

What's wrong with this program?

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
  int *num_ptr = malloc(sizeof(int));
  if (!num_ptr) {
    printf("Unable to allocate.\n");
    exit(1);
  }
  *num_ptr = 331;
  printf("%d rocks!\n", *num_ptr);
  return 0;
}
```

```c
free(num_ptr);
```

(does this bug actually matter for this program?)

You **cannot** understand a C program if you don't know **rules 0**, **1**, and **2**.

Recap & Next Class

Today we learned:

More course mechanics

Feedback

Some C

Next class:

Cuckoo’s Egg discussion

More C