CS 253: Introduction to Systems Programming

Instructor

Instructor:	Jim Buffenbarger
Office:	CCP-359
Email:	`buff@cs.boisestate.edu`
Phone:	208-426-3567
WWW:	`http://cs.boisestate.edu/~buff`

BSU COVID-19 Statement

Meetings

Lectures:	TuTh 12:00–1:15	Zoom
Office hours:	MoWe 3:30–4:30	Zoom
	by appointment	Zoom

Our Teaching Assistant / Grader is Arjun Shukla:

```
arjunshukla@u.boisestate.edu
```

CS Tutoring Center office hours can be found at:

```
onyx:~jbuffenb/classes/253/pub/TutorOfficeHours
```

Catalog Description

Structure of C programs: functions, scope, arrays, structures, pointers, and run-time memory management. Generic programming techniques. Introduction to build systems, debugging techniques, version control, shell scripting, and process management. Security vulnerabilities, buffer overflow, and dynamic memory analysis. Basic systems programming including topics such as streams, buffers, pipes, system calls, processes, threads, and libraries for Linux and Microsoft Windows.

COREQ: CS-HU 250. PREREQ: CS 221.

Goals/Objectives

- Introduction
  - Identify and understand the course materials, activities, and assessments
  - Review programming language concepts
  - C program organization and the main function
  - Automatic variables and the int data type
  - Control statements
  - I/O functions
- Software Development and Tools
  - Requirements analysis
  - Makefiles, make, libraries, C pre-processor, compiler, assembler and linker
  - Crafting test cases to exercise a product's happy and exceptional paths
  - Testing
  - Finding memory leaks
  - Debugging with gdb
- C Programming
  - Coding
  - Introduce the C Standard Library APIs
  - Apply safe string processing techniques in C
  - Identify how arrays work in C
  - Understand conditional compilation and header file guards
- Advanced Programming
  - Reference and pointer variables
  - Introduce the C memory management (malloc, free)
  - The runtime stack, function calls, parameters and returned values
  - Security and the buffer overflow problem
  - Portability topics for popular operating systems
- Processes
  - Introduce the concept of a process
  - Introduce fork, wait and exec

  - Introduce the memory layout of an executable (text, data, stack, heap)

- Signals
  - Introduce the concept of a signal
  - Introduce the concept of asynchronous programming.
  - Apply signal handling
- Pipes
  - Introduce the concept of a pipe
  - Apply pipes between a parent and child process
- Libraries
  - Shared vs. static libraries
  - Building static libraries
- Threads
  - Introduce concurrency with threading in Java
  - Introduce the concept of threading in C with the pthread library
  - Shared data, critical sections and race conditions
  - The producer-consumer problem
  - Mutual exclusion with monitors and mutexes

Textbooks

The textbooks are:

- *The C Programming Language* (K&R), by Brian Kernighan and Dennis Ritchie. Prentice Hall, second edition, 1988. [or, any other C Programming book/guide/tutorial]
- *Managing Projects with GNU Make*, by Robert Meclenburg, 2005. [in our `pub` directory]
- *Advanced Bash-Scripting Guide*, by Mendel Cooper, 2014. [in our `pub` directory]
- *The Art of Unix Programming*, by Eric Steven Raymond, 2003. [in our `pub` directory]

Reading schedule:

- C Programming: Read the equivalent of K&R chapters 1–6, one chapter per lecture. This is the reading shown in the schedule, at the end of this syllabus.
- Make: Read chapters 1–3, one chapter per week.
- Bash: Use this as a reference.
- The Art of Unix Programming: Read chapters 1, 4, and 5. Do this over the first half of the semester.

Other Course Material

This syllabus, lecture slides, assignments, and other material is available on the computers in the Computer Science Labs (CCP-240, CCP-241, and CCP-242), served by `onyx.boisestate.edu`, which is remotely accessible, via Secure Shell (SSH). It is *not* on the WWW, Blackboard, or elsewhere. It is in what is called our "`pub`" directory:

```
onyx:~jbuffenb/classes/253/pub
```

Grading

At the end of the course, a letter grade is assigned to each student according to rank among classmates, which is determined from numerical scores assigned for performance of these activities:

Activity	*Weight*
Homework	50%
Exam	25%
Final	25%

Homework is due at 11:59PM, Mountain Time, on the day it is due. Late work is not accepted. To submit your solution to an assignment, login to a lab computer, change to the directory containing the files you want to submit, and execute:

```
submit jbuffenb class assignment
```

For example:

```
submit jbuffenb cs101 hw1
```

The `submit` program has a nice `man` page.

When you submit a program, include: the source code, sample input data, and its corresponding results.

Scores are posted near my office, as they become available. You are encouraged to check your scores to ensure they are recorded properly. If you feel that a grading mistake has been made, contact me within two weeks of the date that work is returned.

Homework

Several programs are assigned. They require analysis, design, implementation, debugging, and testing. Programs are compiled, tested, and graded on an equivalent to `onyx.boisestate.edu`, using the GNU/Linux toolchain (e.g. `gcc`). You can develop your homework in a different environment (e.g., your laptop), but you must then port it to `onyx`.

Exam and Final

An exam and a final are administered. These are in-class, open-note, and open-textbook (but no other books) tests.

Makeup examinations are not normally administered.

Source-Code Documentation

Good documentation and programming style is very important. Your programs must demonstrate these qualities for full credit. Good documentation and programming style includes:

- heading comments giving: author, date, class, and description
- function/procedure comments giving description of: purpose, parameters, and return value
- other comments where clarification of source code is needed
- proper and consistent indentation
- proper structure and modularity

For more information, and examples, see:

`www.cs.swarthmore.edu/~newhall/unixhelp/c_codestyle.html`

Academic Integrity

The University's goal is to foster an intellectual atmosphere that produces educated, literate people. Because cheating and plagiarism are at odds with that goal, those actions shall not be tolerated in any form. Academic dishonesty includes assisting a student to cheat, plagiarize, or commit any act of academic dishonesty. Plagiarism occurs when a person tries to represent another person's work as his or her own or borrows directly from another person's work without proper documentation.

If a student engages in academic dishonesty, the student may be dismissed from the class and may receive a failing grade. Other penalties may include suspension or expulsion from the University.

Much more information about academic integrity, including examples of academic dishonesty, is at:

http://cs.boisestate.edu/~buff/files/www-integrity.pdf

If you are unsure about a particular behavior, ask your instructor.

Labs and Safety

Each student receives an account on the cluster of computers in the Computer Science Labs: CCP-240, CCP-241, and CCP-242. The cluster comprises a server named `onyx.boisestate.edu` and a set of nodes with shared home directories. It is remotely accessible, via SSH. The cluster runs the Linux and Windows operating systems, via VMware.

Physical access requires building and room access. After-hours building access, and all-hours room access, require an authenticated proximity-type student-identification card.

You are responsible for understanding and obeying lab rules:

http://coen.boisestate.edu/its/lab-rules

The health and safety of all members of our academic community is very important. While computer science is a relatively safe science/engineering discipline, dangers exist, and we should be prepared for them. Basically, call 911 to report an emergency. Beyond that, please take a moment to review this common-sense information:

http://coen.boisestate.edu/cs/safetydocument

Schedule

Week	*Date*	*Topic*	*Assigned*	*Due*	*Reading*
1	Aug 25 Tue	Introduction			1
	Aug 27 Thu	C Programming			2
2	Sep 01 Tue		HW1		3
	Sep 03 Thu				4
3	Sep 08 Tue				5
	Sep 10 Thu			HW1	6
4	Sep 15 Tue		HW2		
	Sep 17 Thu				
5	Sep 22 Tue				
	Sep 24 Thu				
6	Sep 29 Tue	Bash Programming			
	Oct 01 Thu		HW3	HW2	
7	Oct 06 Tue				
	Oct 08 Thu				
8	Oct 13 Tue	Make and Makefiles			
	Oct 15 Thu			HW3	
9	Oct 20 Tue	Exam			
	Oct 22 Thu		HW4		
10	Oct 27 Tue	Unix/GCC Tool Chain			
	Oct 29 Thu				
11	Nov 03 Tue				
	Nov 05 Thu		HW5	HW4	
12	Nov 10 Tue	Unix Programming			
	Nov 12 Thu				
13	Nov 17 Tue				
	Nov 19 Thu		HW6	HW5	
14	Nov 24 Tue	Thanksgiving			
	Nov 26 Thu	Thanksgiving			
15	Dec 01 Tue				
	Dec 03 Thu				
16	Dec 08 Tue				
	Dec 10 Thu			HW6	
17	Dec 15 Tue	Final: 12:00-2:00			