15-213/18-243: Introduction to Computer Systems

Randy Bryant and Dave O'Hallaron

Carnegie Mellon University
Fall 2010

1 Organization

Class Web page: `http://www.cs.cmu.edu/~213`

Electronic copies of class handouts and lectures can be found on the class Web page.

Teaching staff email address: `15-213-staff@cs.cmu.edu`

Please send email to this address whenever you have questions about the course. Don't send mail to individual staff members except to schedule one-on-one meetings. Any emails you send to this address will be received by all members of the teaching staff. Using it allows us to give you the fastest and most consistent responses to your questions. We will not be using Blackboard or any other message board.

Instructors:

Randy Bryant
randy.bryant@cs.cmu.edu
GHC 5113, 412-268-8821
Office hours: See class Web page

Dave O'Hallaron
droh@cs.cmu.edu
GHC 9125, 412-268-8199
Office hours: See class Web page

TAs:

Joe Appel
jsappel@andrew.cmu.edu

Tom Conerly
tconerly@andrew.cmu.edu

Taerim Kim
taerimk@cmu.edu

Ted Martin
tdmartin@andrew.cmu.edu

Hunter Pitelka
hpitelka@andrew.cmu.edu

Hari Seshadri
sseshadr@andrew.cmu.edu

Lucas Tan
btan@andrew.cmu.edu

Yi Zhuang
yzhuang@andrew.cmu.edu

Lectures: Tuesday and Thursday 1:30–2:50pm, DH 2315

Recitations:

A	Mon	10:30–11:20	WEH 5310	Tom Conerly
B	Mon	10:30–11:20	WEH 6423	Joe Appel
C	Mon	11:30–12:20	WEH 5310	Srihari Seshadri
D	Mon	12:30–1:20	WEH 5302	Hunter Pitelka
E	Mon	1:30–2:20	PH 226B	Lucas Tan
F	Mon	1:30–2:20	DH 1211	Yi Zhuang
G	Mon	2:30–3:20	WEH 5310	Taerim Kim
H	Mon	3:30–4:20	PH 125C	Teddy Martin

TA Office Hours:
See class Web page for office hours.

2 Objectives

Our aim in 15-213 is to help you become a better programmer by teaching you the basic concepts underlying all computer systems. We want you to learn what really happens when your programs run, so that when things go wrong (as they always do) you will have the intellectual tools to solve the problem.

Why do you need to understand computer systems if you do all of your programming in high level languages? In most of computer science, we're pushed to make abstractions and stay within their frameworks. But, any abstraction ignores effects that can become critical. As an analogy, Newtonian mechanics ignores relativistic effects. The Newtonian abstraction is completely appropriate for bodies moving at less than $0.1c$, but higher speeds require working at a greater level of detail.

Oversimplifying matters somewhat, our $21x$ sequence works as follows: 211 is based on a simplified model of program execution. 212 builds further layers of abstraction. 213 introduces greater detail about system behavior and operation. This greater detail is needed for optimizing program performance, for working within the finite memory and word size constraints of computers, and for systems-level programming.

The following "realities" are some of the major areas where the abstractions we teach in 211/212 break down:

1. *Int's are not integers, Float's are not reals.* Our finite representations of numbers have significant limitations, and because of these limitations we sometimes have to think in terms of bit-level representations.

2. *You've got to know assembly language.* Even if you never write programs in assembly, The behavior of a program cannot be understood sometimes purely based on the abstraction of a high-level language. Further, understanding the effects of bugs requires familiarity with the machine-level model.

3. *Memory matters.* Computer memory is not unbounded. It must be allocated and managed. Memory referencing errors are especially pernicious. An erroneous updating of one object can cause a change in some logically unrelated object. Also, the combination of caching and virtual memory provides the functionality of a uniform unbounded address space, but not the performance.

4. *There is more to performance than asymptotic complexity.* Constant factors also matter. There are systematic ways to evaluate and improve program performance.

5. *Computers do more than execute instructions.* They also need to get data in and out and they interact with other systems over networks.

By the end of the course, you will understand these "realities" in some detail. As a result, you will be prepared to take any of the upper-level systems classes at Carnegie Mellon (both CS and ECE). Even more important, you will have learned skills and knowledge that will help you throughout your career.

3 Textbook

The primary textbook for the course is

> Randal E. Bryant and David R. O'Hallaron, *Computer Systems: A Programmer's Perspective, Second Edition (CS:APP2e)*, Prentice Hall, 2011.

Please make sure you have the Second Edition, which is significantly different from the First Edition published in 2003. In addition, we require you to have the following reference book on the C programming language:

> Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language, Second Edition*, Prentice Hall, 1988.

This is the classic *K & R* book, the standard against which all reference manuals are compared. It is an essential part of every computer scientist's library.

4 Course Organization

Your participation in the course will involve five forms of activity:

1. Attending the lectures.
2. Preparing for and participating in the recitations.
3. Doing laboratory assignments.
4. Reading the text.
5. Taking exams

Attendance will not be taken at the lectures or recitation sections. You will be considered responsible for all material presented at the lectures and recitations.

Lectures will cover higher-level concepts. Recitations will be more applied, covering important "how-to's", especially in using tools that will help you do the labs. In addition, the recitations will help clarify lecture topics and describe exam coverage.

The textbook contains both *practice problems* within the chapter text and *homework problems* at the end of each chapter. The intention is that you work on the practice problems as you are reading the book. The answers to these problems are at the end of each chapter. Our experience has been that trying out the concepts on simple examples helps make the ideas more concrete. In addition, the schedule (on the class Web page) shows specific homework problems with each lecture topic. The intention is that you try these out and discuss them in the next recitation. You will find that you will get much more out of recitation if you have done some advance preparation.

The only graded assignments in this class will be a set of six labs. Some of these are fairly short, requiring just one week, while others are more ambitious, requiring several weeks.

5 Getting Help

For all communication with the teaching staff, please send email to `15-213-staff@cs.cmu.edu`.

We will use the class website (`http://www.cs.cmu.edu/~213`) as the central repository for all information about the class.

The lab assignments and class message board are offered through a Web service written by Hunter Pitelka and Prof. David O'Hallaron called *Autolab*. See the Autolab Web page at `http://autolab.cs.cmu.edu` for more information.

If you want to talk to a staff member in person, the posted office hours are the best opportunity, as the they represent times when we guarantee that we will be in the location identified. If a meeting is needed outside of the 10 office hours, please use email to arrange a time. (Note that `15-213-staff@cs.cmu.edu` reaches all 10 teaching staff, maximizing your odds of a rapid turnaround.)

Prof. Bryant is glad to meet with students, but he's also got a lot of other meetings. The best way to meet with him is to set up an appointment through his assistant, June Fischerkeller, GHC 5111, x8-7884, (`jfische@cs.cmu.edu`).

6 Policies

Working Alone on Assignments

You will work on all assignments by yourself.

Handing in Assignments

All assignments are due at 11:59pm (one minute before midnight) on the specified due date. All handins are electronic using the Autolab system. You may handin in as often you like, with your most recent handin

counting for credit.

Handing in Late Assignments

The penalty for late assignments is 20% per day. Each student will receive a budget of three *grace days* for the course. These grace days are provided to allow you to cope with most emergencies that prevent completing a lab on time, including computer problems, a cold, getting stuck at the airport, etc. Here is how grace days work:

- You will receive at most one grace day for each assignment. Grace days are applied automatically until you run out. You cannot defer or bank grace days.
- If your last handin is one day late, and you have some remaining grace days, then you will receive full credit for the lab and automatically spend one grace day. For example, if an assignment is due at 11:59pm on Thursday and your last handin is noon on Friday, then you will receive full credit and spend one grace day.
- Once you have spent a grace day, then you will receive a penalty of 20% for each subsequent late day. For example, if an assignment is due at 11:59pm on Thursday and your last handin is noon on Saturday, then you will spend one grace day and be penalized 20%. If your last handin is noon on Sunday, then you will spend one grace day and be penalized 40%.
- Handins will not be accepted after the *end date* of the lab, which is typically three days after the due date.

Grace days are a tool to allow you to manage your time in the face of personal issues and to help smooth out burstiness in assignment due dates across classes. They are for when you are sick, when a short-term emergency situation arises, when you have too many deadlines all at once, etc. Except for serious persistent personal issues (see below), you should not anticipate additional deadline leniency. We recommend that you conserve your grace days, saving them for true emergencies and time pressures at the end of the term.

Dealing with Serious Persistent Personal Issues

We hope that everyone in 15-213 will remain happy and healthy. But, if you have a serious persistent personal issue, such as being hospitalized for an extended period or needing to leave the country for a family matter, please talk to your academic advisor as soon as possible. Such issues consistently affect one's ability to succeed in all classes, rather than just 15-213, and the academic advisors are equipped to coordinate plans for dealing with them. We will cooperate with such plans, but we cannot construct them independently of the academic advisors.

Requesting a Re-Grade for an Assignment or an Exam

After each exam and lab assignment is graded, your score will be posted on the Autolab gradebook. We will make the utmost effort to be fair and consistent in our grading. But, we are human. If you believe that you did not receive appropriate credit for an assignment or an exam, you may request a re-grade as follows:

- Submit your request *in writing* within seven calendar days of when the grade notification is sent to you via email, explaining in detail why you think that there was a mistake in the grading. Please note that verbal requests will not be processed; requests must be in writing.
- These requests should be submitted to `15-213-staff@cs.cmu.edu`, whether for an assignment or for an exam. In the case of exams, the exam in question should be hand-delivered to the course assistant together with a printed copy of the email request.
- When you submit a request for a re-grade, the entire assignment or exam may be re-graded (not just the parts that you specify). Your grade may go up or down (or stay the same) as a result of the re-grade request.

Your request will be processed off-line, and we will respond to your request as quickly as possible (typically within a week). This re-grade policy is designed to correct legitimate mistakes in grading, while discouraging frivolous re-grade requests (for the sake of being fair and consistent across the entire class).

Final Grade Assignment

Each student will receive a numeric score for the course, based on a weighted average of the following:

- **Assignments (50%):** There are a total of seven assignments (labs), which will count a combined total of 50% of your score. Assignments have different weightings, based on our perception of the relative effort required. See the class Web page for the assignment weightings.
- **Exams (50%):** There will be two in-class exams, each counting 12.5%, plus a final counting 25%.

Grades for the course will be determined by a method that combines both curving and absolute standards. The total score will be plotted as a histogram. Cutoff points are determined by examining the quality of work by students on the borderlines. Individual cases, especially those near the cutoff points may be adjusted upward or downward based on factors such as attendance, class participation, improvement throughout the course, final exam performance, and special circumstances.

Cheating

Each exam and lab assignment must be the sole work of the student turning it in. Assignments will be closely monitored by automatic cheat checkers, including comparing turned-in code to the work of students from the same and previous semesters, and students may be asked to explain any suspicious similarities. These cheat checkers are very effective, having been refined over years of research, and they are not fooled by attempts to mask copying of code. Please don't try your luck.

The usual penalty for cheating is to be removed from the course with a failing grade. The University also places a record of the incident in the student's permanent record.

The following are guidelines on what non-exam collaboration is authorized and what is not:

What is Cheating?

- *Sharing code or other electronic files*: either by copying, retyping, looking at, or supplying a copy of a file from this or a previous semester. Be sure to store your work in protected directories, and log off when you leave an open cluster, to prevent others from copying your work without your explicit assistance.
- *Sharing written assignments or exams*: Looking at, copying, or supplying an assignment or exam.
- *Using other's code.* Using code from this or previous offerings of 15-213, from courses at other institutions, or from any other non-213 source (e.g., software found on the Internet).
- *Looking at other's code.* Although mentioned above, it bears repeating. Looking at other students' code or allowing others to look at yours is cheating. There is no notion of looking "too much," since no looking is allowed at all.

What is NOT Cheating?

- Clarifying ambiguities or vague points in class handouts or textbooks.
- Helping others use the computer systems, networks, compilers, debuggers, profilers, or other system facilities.
- Helping others with high-level design issues.
- Helping others with high-level (not code-based) debugging.
- Using code from the CS:APP website or from the class Web pages.

Be sure to store your work in protected directories, and log off when you leave an open cluster, to prevent others from copying your work without your explicit assistance.

7 Facilities: Intel Computer Systems Cluster

Intel Corp. has generously donated a cluster of 21 Linux-based 64-bit multicore Nehalem servers, specifically for 15-213/18-243, that we will use for all labs and assignments. The class Web page has details.

8 Class Schedule

Please see the schedule maintained on the class Web page for information about lectures, reading assignments, suggested homework problems, lab start and end dates, and the lecturer for each class. The reading assignments are all from the CS:APP2e book.