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

Bill Nace and Gregory Kesden

Carnegie Mellon University
Spring 2010

1 Organization

Class Web Page: `http://www.cs.cmu.edu/~213`
Note: Electronic copies of class handouts and lectures can be found on the class web page.

Class Message Board: `http://autolab.cs.cmu.edu`
Note: This is the only message board that the 15-213/18-243 staff will be monitoring. We will not be using Andrew or Blackboard message boards for this class.

Contacting Teaching Staff: Send email to `15-213-staff@cs.cmu.edu`. This will go to all of the teaching staff.
Note: Please use 15-213-staff@cs.cmu.edu for all questions and concerns, rather than single-person emails, other than scheduling of one-on-one meetings. You will get faster answers, and we will better be able to provide consistency.

Instructors:

Bill Nace
wnace@cmu.edu
HH D207, x8-7027
Office hours: Wednesdays 2:00-4:30

Gregory Kesden
gkesden@cs.cmu.edu
GHC 7711, x8-1590
`http://www.cs.cmu.edu/~gkesden/schedule.html`

TAs:

Dan Burrows
dburrows@andrew.cmu.edu

Timothy Douglas
tdouglas@andrew.cmu.edu

Jason Franklin
jfrankli@andrew.cmu.edu

Alex Gartrel
agartrel@andrew.cmu.edu

Ted Martin
tdmartin@andrew.cmu.edu

Mike Mu
machongm@andrew.cmu.edu

Hunter Pitelka
hpitelka@andrew.cmu.edu

Josh Primero
jprimero@andrew.cmu.edu

Tom Tuttle
ttuttle@andrew.cmu.edu

Joel Feinstein
jnfeinst@andrew.cmu.edu

Lectures:

- Lecture 1: Tuesday, Thursday 6:30–7:50pm, GHC 4401
- Lecture 2: Tuesday, Thursday 1:30–2:50pm, GHC 4401

Recitations:

A	Mon	9:30–10:20	WEH 5302
B	Mon	10:30–11:20	WEH 5302
C	Mon	11:30–12:20	WEH 5302
D	Mon	12:30–1:20	WEH 5302
E	Mon	1:30–2:20	WEH 5302
F	Mon	11:30–12:20	WEH 6423
G	Mon	12:30–1:20	WEH 6423
H	Mon	1:30–2:20	WEH 6423
I	Mon	2:30–3:20	WEH 6423
J	Mon	3:30–4:20	WEH 6423

TA Office Hours:
Sundays - Thursdays, 6:00PM - 9:00PM in the Wean Hall 5207 cluster.

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*, Prentice Hall, 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. Laboratory assignments.

4. Reading the text.

5. 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 right 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 (at the end of this document and 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 urgent 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`) and class message board (*Autolab*) as the central repositories for all information about the class. Using these resources, you can:

- Obtain copies of any handouts or assignments. This is especially useful if you miss class or you lose your copy.
- Find links to any electronic data you need for your assignments
- Read clarifications and changes made to any assignments, schedules, or policies.
- Post messages messages to make queries about the course, specific labs, or exams. (And, of course, see responses to such posts.)

The lab assignments and class message board are offered through a Web service written by 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.)

6 Policies

Working Alone on Assignments

You will work on all assignments by yourself, unless notified otherwise in the assignment description.

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.

Handing in Late Assignments

Each student will receive a budget of four *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:

- If you hand in an assignment k days late, then you receive full credit for the lab, but you will have spent k of your grace days. For example, if an assignment is due at 11:59pm on Thursday and you hand it in at noon on Saturday, then you will have spent 2 grace days. If you hand it in at 9am on Friday, then you will have spent 1 grace day.
- When you are out of grace days, assignments turned in before the *end date* (see next bullet) will receive a 15% late penalty per day. So, for example, an assignment turned two days late would receive a 0% penalty if you had two or more grace days, a 15% penalty if you had one grace day, and a 30% penalty if you had zero grace days.
- Regardless of the number of grace days you have remaining, handins will not be accepted after the *end date* of the lab, which is typically 3 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, a personalized email will be sent to each of you with your grade (as well as all of your previous grades). 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

To do well in this course, you must do well on both the lab assignments *and* the exams. These two different forms of evaluation test different aspects of the material—how well you understand the underlying concepts (exams) and how well you can put the concepts into practice (labs).

You will receive an overall score, between 0 and 100 points, for the course. This score is based on both your lab and exam averages. The relative weight of the labs and exams is based on a a 60-40 tilt as follows: The *lower* of your exam average and your lab average counts sixty percent, the higher of the two counts fourty-percent. You cannot pass the course unless each of your lab and exam averages are passing. The goal of this grading scheme is to ensure that each student has the necessary knowledge, skill, and experience to succeed in upper-division systems classes.

The final exam counts as much as the two other exams combined. In other words, each exam counts 25-percent of the exam average and the final exam counts 50-percent of the exam average.

The weight of each lab is proportional to the effort required for that lab. In oorder to allow proper tuning of the course, the exact weight of each lab is not determined until very late in the semester. The weight of an individual lab is likely to vary from 10-percent of the lab average for a simple, short lab, to 25-percent of the lab average for a long, complex lab.

Given the overall score grades for the course will be determined by a method that combines both curving and absolute standards. The overall scores 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. We also place 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.
- *Sharing written assignments*: Looking at, copying, or supplying an assignment.
- *Using other's code.* Using code from this or previous offerings of 15-213, as well as from courses at other institutions.

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 course web pages.

Be sure to store your work in protected directories.

7 Facilities: Intel Computer Systems Cluster

Intel has generously donated a cluster of 15 Linux-based 64-bit Xeon servers, specifically for 15-213, 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:APP book.