

Course Overview

15-213/18-213/15-513: Introduction to Computer Systems
1st Lecture, Sep. 1, 2015

Instructors:

Randal E. Bryant and David R. O'Hallaron

The course that gives CMU its “Zip”!

Overview

- Course theme
- Five realities
- How the course fits into the CS/ECE curriculum
- Academic integrity

Course Theme:

Abstraction Is Good But Don't Forget Reality

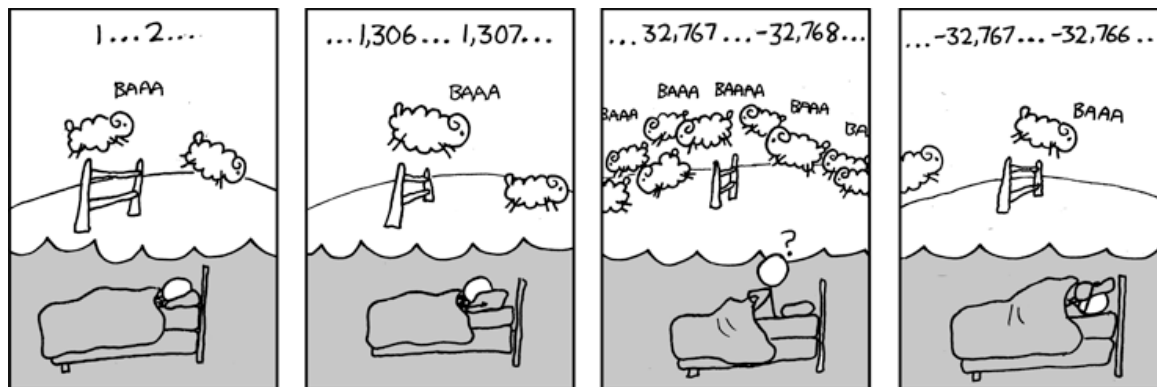
- **Most CS and CE courses emphasize abstraction**
 - Abstract data types
 - Asymptotic analysis
- **These abstractions have limits**
 - Especially in the presence of bugs
 - Need to understand details of underlying implementations
- **Useful outcomes from taking 213**
 - Become more effective programmers
 - Able to find and eliminate bugs efficiently
 - Able to understand and tune for program performance
 - Prepare for later “systems” classes in CS & ECE
 - Compilers, Operating Systems, Networks, Computer Architecture, Embedded Systems, Storage Systems, etc.

Great Reality #1:

Ints are not Integers, Floats are not Reals

■ Example 1: Is $x^2 \geq 0$?

- Float's: Yes!



- Int's:

- 40000 * 40000 → 1600000000
- 50000 * 50000 → ??

■ Example 2: Is $(x + y) + z = x + (y + z)$?

- Unsigned & Signed Int's: Yes!

- Float's:

- $(1e20 + -1e20) + 3.14 \rightarrow 3.14$
- $1e20 + (-1e20 + 3.14) \rightarrow ??$

Computer Arithmetic

■ Does not generate random values

- Arithmetic operations have important mathematical properties

■ Cannot assume all “usual” mathematical properties

- Due to finiteness of representations
- Integer operations satisfy “ring” properties
 - Commutativity, associativity, distributivity
- Floating point operations satisfy “ordering” properties
 - Monotonicity, values of signs

■ Observation

- Need to understand which abstractions apply in which contexts
- Important issues for compiler writers and serious application programmers

Great Reality #2: You've Got to Know Assembly

- **Chances are, you'll never write programs in assembly**
 - Compilers are much better & more patient than you are
- **But: Understanding assembly is key to machine-level execution model**
 - Behavior of programs in presence of bugs
 - High-level language models break down
 - Tuning program performance
 - Understand optimizations done / not done by the compiler
 - Understanding sources of program inefficiency
 - Implementing system software
 - Compiler has machine code as target
 - Operating systems must manage process state
 - Creating / fighting malware
 - x86 assembly is the language of choice!

Great Reality #3: Memory Matters

Random Access Memory Is an Unphysical Abstraction

■ Memory is not unbounded

- It must be allocated and managed
- Many applications are memory dominated

■ Memory referencing bugs especially pernicious

- Effects are distant in both time and space

■ Memory performance is not uniform

- Cache and virtual memory effects can greatly affect program performance
- Adapting program to characteristics of memory system can lead to major speed improvements

Memory Referencing Bug Example

```
typedef struct {
    int a[2];
    double d;
} struct_t;

double fun(int i) {
    volatile struct_t s;
    s.d = 3.14;
    s.a[i] = 1073741824; /* Possibly out of bounds */
    return s.d;
}
```

```
fun(0)    →    3.14
fun(1)    →    3.14
fun(2)    →    3.13999998664856
fun(3)    →    2.000000061035156
fun(4)    →    3.14
fun(6)    →    Segmentation fault
```

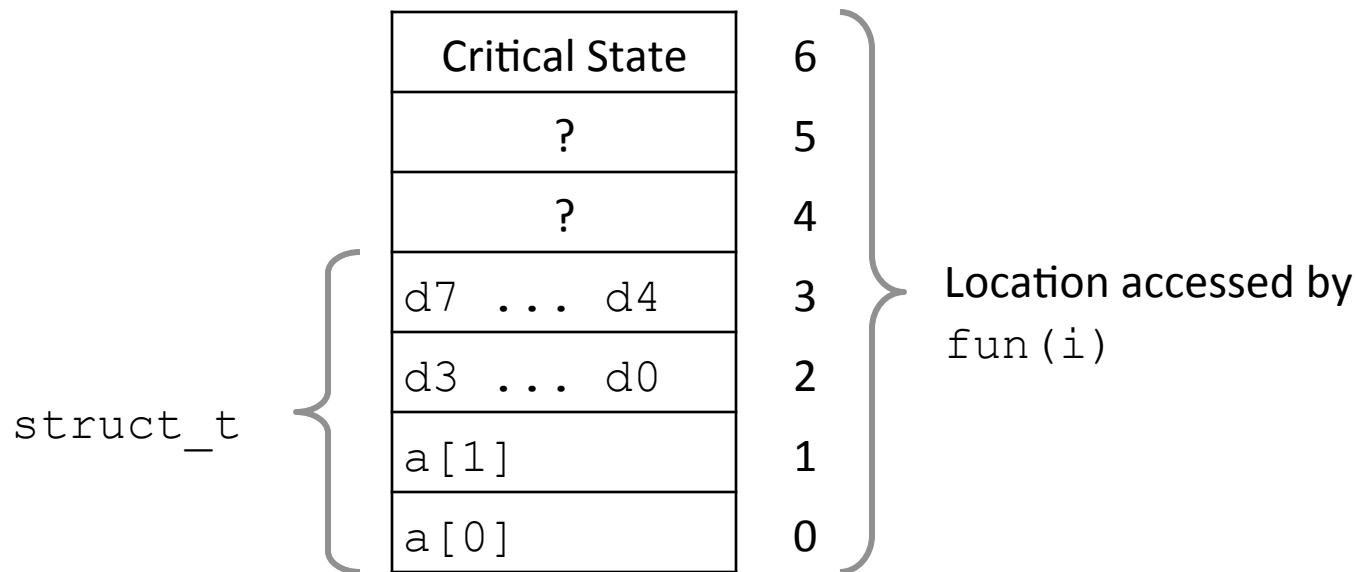
- Result is system specific

Memory Referencing Bug Example

```
typedef struct {
    int a[2];
    double d;
} struct_t;
```

```
fun(0) → 3.14
fun(1) → 3.14
fun(2) → 3.1399998664856
fun(3) → 2.00000061035156
fun(4) → 3.14
fun(6) → Segmentation fault
```

Explanation:



Memory Referencing Errors

■ C and C++ do not provide any memory protection

- Out of bounds array references
- Invalid pointer values
- Abuses of malloc/free

■ Can lead to nasty bugs

- Whether or not bug has any effect depends on system and compiler
- Action at a distance
 - Corrupted object logically unrelated to one being accessed
 - Effect of bug may be first observed long after it is generated

■ How can I deal with this?

- Program in Java, Ruby, Python, ML, ...
- Understand what possible interactions may occur
- Use or develop tools to detect referencing errors (e.g. Valgrind)

Great Reality #4: There's more to performance than asymptotic complexity

- **Constant factors matter too!**
- **And even exact op count does not predict performance**
 - Easily see 10:1 performance range depending on how code written
 - Must optimize at multiple levels: algorithm, data representations, procedures, and loops
- **Must understand system to optimize performance**
 - How programs compiled and executed
 - How to measure program performance and identify bottlenecks
 - How to improve performance without destroying code modularity and generality

Memory System Performance Example

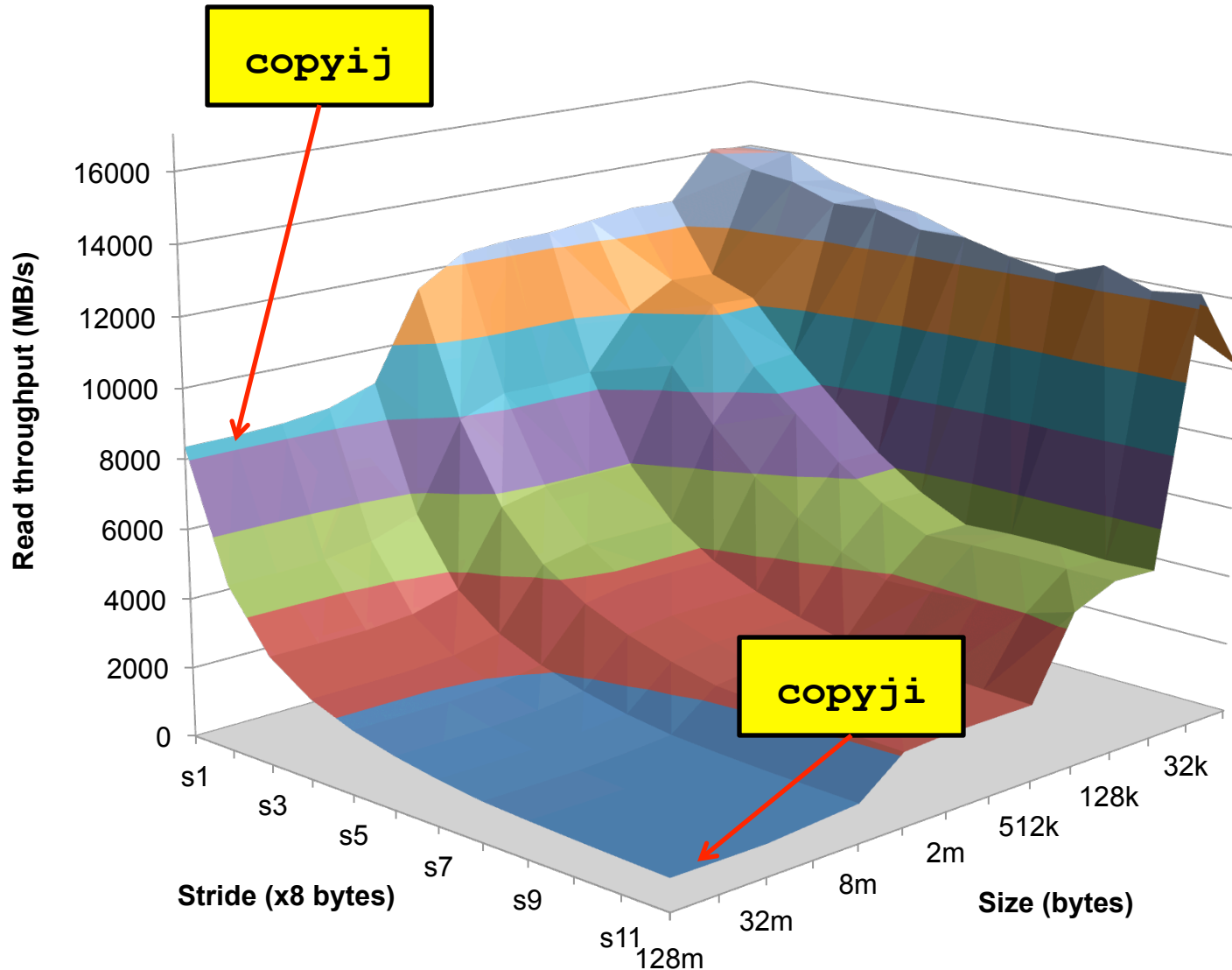
```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (j = 0; j < 2048; j++)
        for (i = 0; i < 2048; i++)
            dst[i][j] = src[i][j];
}
```

4.3ms 2.0 GHz Intel Core i7 Haswell 81.8ms

- Hierarchical memory organization
- Performance depends on access patterns
 - Including how step through multi-dimensional array

Why The Performance Differs

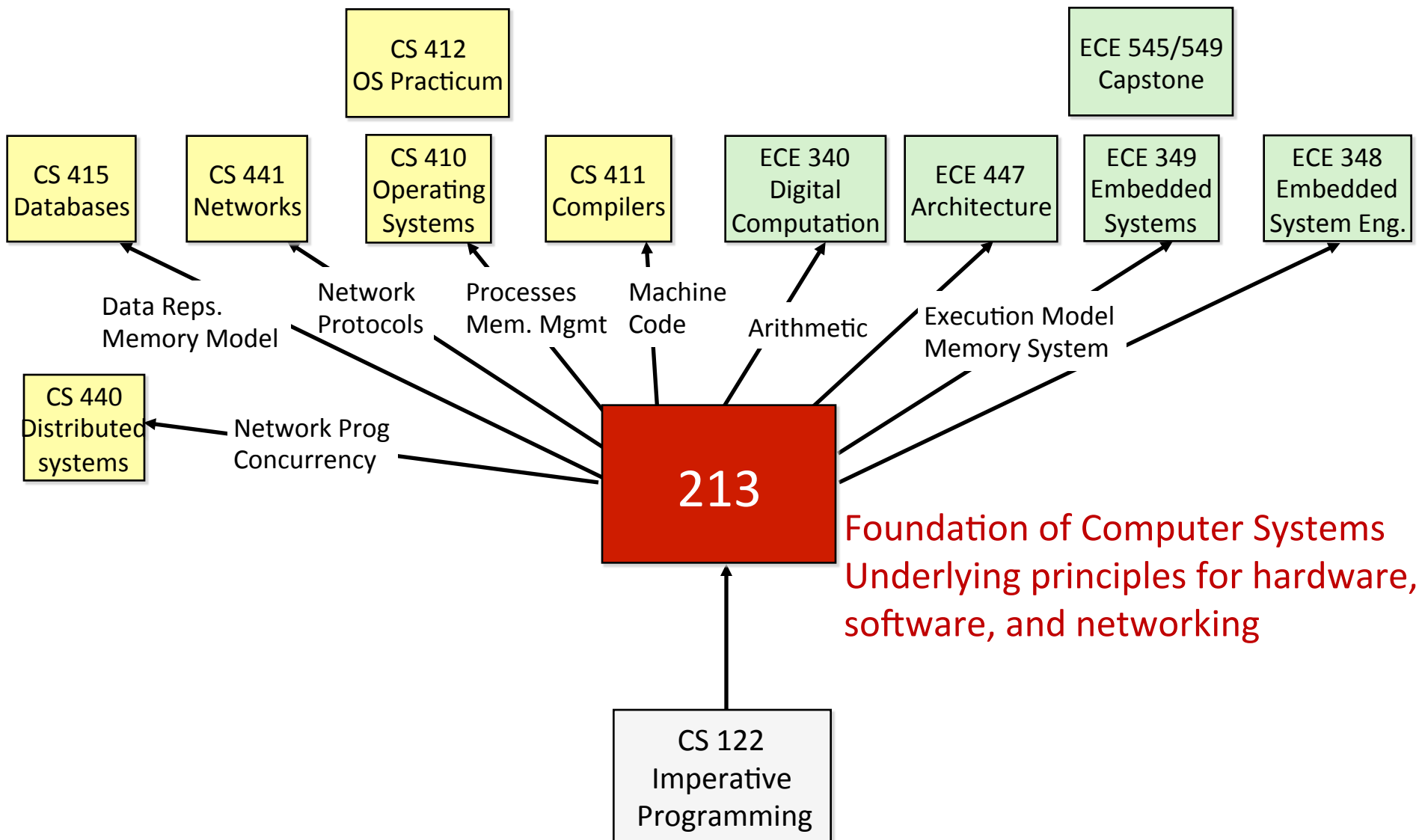


Great Reality #5:

Computers do more than execute programs

- **They need to get data in and out**
 - I/O system critical to program reliability and performance
- **They communicate with each other over networks**
 - Many system-level issues arise in presence of network
 - Concurrent operations by autonomous processes
 - Coping with unreliable media
 - Cross platform compatibility
 - Complex performance issues

Role within CS/ECE Curriculum



Course Perspective

- Most Systems Courses are Builder-Centric
 - Computer Architecture
 - Design pipelined processor in Verilog
 - Operating Systems
 - Implement sample portions of operating system
 - Compilers
 - Write compiler for simple language
 - Networking
 - Implement and simulate network protocols

Instructors

Randy Bryant



Dave O'Hallaron



15-513 vs 15-213 and 18-213

- There is enormous demand from MS students for 213.
 - In the past, many MS students could not get in. Here's our solution...
- 15-213 and 18-213 are for undergrads only.
 - Undergrads will attend lectures and recitations in person, as usual
- 15-513 is for grad students only.
 - In order to accommodate the volume of students, grad students do not attend recitation and lecture in person.
 - We will videotape each lecture and recitation and post them afterward on the course Web site (<http://www.cs.cmu.edu/~213>)
- For help, all students have equal access to the TA office hours and staff mailing list.
- All students do the same assignments and exams.

Cheating: Description

- **Please pay close attention, especially if this is your first semester at CMU**

- What is cheating?
 - Sharing code: by copying, retyping, **looking at**, or supplying a file
 - Describing: verbal description of code from one person to another.
 - Coaching: helping your friend to write a lab, line by line
 - Searching the Web for solutions
 - Copying code from a previous course or online solution
 - You are only allowed to use code we supply, or from the CS:APP website

- What is NOT cheating?
 - Explaining how to use systems or tools
 - Helping others with high-level design issues

- See the course syllabus for details.
 - Ignorance is not an excuse

Cheating: Consequences

- Penalty for cheating:
 - Removal from course with failing grade (no exceptions!)
 - Permanent mark on your record
 - Your instructors' personal contempt

- Detection of cheating:
 - We have sophisticated tools for detecting code plagiarism
 - Last Fall, 25 students were caught cheating and failed the course.
 - Some were expelled from the University

- Don't do it!
 - Start early
 - Ask the staff for help when you get stuck

Textbooks

- Randal E. Bryant and David R. O'Hallaron,
 - *Computer Systems: A Programmer's Perspective*, **Third Edition** (CS:APP3e), Pearson, 2016
 - <http://csapp.cs.cmu.edu>
 - This book really matters for the course!
 - How to solve labs
 - Practice problems typical of exam problems

- Brian Kernighan and Dennis Ritchie,
 - *The C Programming Language*, Second Edition, Prentice Hall, 1988
 - Still the best book about C, from the originators

Course Components

■ Lectures

- Higher level concepts

■ Recitations

- Applied concepts, important tools and skills for labs, clarification of lectures, exam coverage

■ Labs (7)

- The heart of the course
- 1-2 weeks each
- Provide in-depth understanding of an aspect of systems
- Programming and measurement

■ Exams (midterm + final)

- Test your understanding of concepts & mathematical principles

Getting Help

- Class Web page: <http://www.cs.cmu.edu/~213>
 - Complete schedule of lectures, exams, and assignments
 - Copies of lectures, assignments, exams, solutions
 - Clarifications to assignments

- Blackboard and Piazza
 - We won't be using Blackboard or Piazza for the course

Getting Help

- Staff mailing list: **15-213-staff@cs.cmu.edu**
 - Use this for all communication with the teaching staff
 - Always CC staff mailing list during email exchanges
 - Send email to individual instructors only to schedule appointments
- Office hours (starting Tue Sep 8):
 - SMTWRF, 5:45-8:30pm, WeH 5207
- 1:1 Appointments
 - You can schedule 1:1 appointments with any of the teaching staff

Policies: Labs And Exams

- Work groups
 - You must work alone on all lab assignments
- Handins
 - Labs due at 11:59pm on Tues or Thurs
 - Electronic handins using **Autolab** (no exceptions!)
- Exams
 - Exams will be online in network-isolated clusters
 - Held over multiple days. Self-scheduled; just sign up!
- Appealing grades
 - In **writing** to Prof O'Hallaron within 7 days of completion of grading
 - Follow formal procedure described in syllabus

Facilities

- Labs will use the Intel Computer Systems Cluster
 - The “shark machines”
 - `linux> ssh shark.ics.cs.cmu.edu`

 - 21 servers donated by Intel for 213
 - 10 student machines (for student logins)
 - 1 head node (for Autolab server and instructor logins)
 - 10 grading machines (for autograding)
 - Each server: iCore 7: 8 Nehalem cores, 32 GB DRAM, RHEL 6.1
 - Rack-mounted in Gates machine room
 - Login using your Andrew ID and password
- Getting help with the cluster machines:
 - Please direct questions to staff mailing list

Timeliness

■ Grace days

- **5 grace days** for the semester
- Limit of **2 grace days** per lab used **automatically**
- Covers scheduling crunch, out-of-town trips, illnesses, minor setbacks
- Save them until late in the term!

■ Lateness penalties

- Once grace day(s) used up, get penalized **15% per day**
- No handins later than **3 days after due date**

■ Catastrophic events

- Major illness, death in family, ...
- Formulate a plan (with your academic advisor) to get back on track

■ Advice

- Once you start running late, it's really hard to catch up

Other Rules of the Lecture Hall

- Laptops: permitted

- Electronic communications: **forbidden**
 - No email, instant messaging, cell phone calls, etc

- Presence in lectures, recitations: voluntary, recommended

- No recordings of ANY KIND

Policies: Grading

- Exams (50%): midterm (20%), final (30%)
- Labs (50%): weighted according to effort
- Final grades based on a straight scale.

Programs and Data

■ Topics

- Bits operations, arithmetic, assembly language programs
- Representation of C control and data structures
- Includes aspects of architecture and compilers

■ Assignments

- L1 (datalab): Manipulating bits
- L2 (bomblab): Defusing a binary bomb
- L3 (attacklab): The basics of code injection attacks

The Memory Hierarchy

■ Topics

- Memory technology, memory hierarchy, caches, disks, locality
- Includes aspects of architecture and OS

■ Assignments

- L4 (cachelab): Building a cache simulator and optimizing for locality.
 - Learn how to exploit locality in your programs.

Exceptional Control Flow

■ Topics

- Hardware exceptions, processes, process control, Unix signals, nonlocal jumps
- Includes aspects of compilers, OS, and architecture

■ Assignments

- L5 (tshlab): Writing your own Unix shell.
 - A first introduction to concurrency

Virtual Memory

■ Topics

- Virtual memory, address translation, dynamic storage allocation
- Includes aspects of architecture and OS

■ Assignments

- L6 (malloclab): Writing your own malloc package
 - Get a real feel for systems-level programming

Networking, and Concurrency

■ Topics

- High level and low-level I/O, network programming
- Internet services, Web servers
- concurrency, concurrent server design, threads
- I/O multiplexing with select
- Includes aspects of networking, OS, and architecture

■ Assignments

- L7 (proxylab): Writing your own Web proxy
 - Learn network programming and more about concurrency and synchronization.

Lab Rationale

- Each lab has a well-defined goal such as solving a puzzle or winning a contest
- Doing the lab should result in new skills and concepts
- We try to use competition in a fun and healthy way
 - Set a reasonable threshold for full credit
 - Post intermediate results (anonymized) on Autolab scoreboard for glory!

Autolab (<https://autolab.cs.cmu.edu>)

- Labs are provided by the CMU Autolab system
 - Project page: <http://autolab.cs.cmu.edu>
 - Developed by CMU faculty and students
 - Key ideas: Autograding and Scoreboards
 - **Autograding:** Providing you with instant feedback.
 - **Scoreboards:** Real-time, rank-ordered, and anonymous summary.
 - Used by over 3,000 students each semester
- With Autolab you can use your Web browser to:
 - Download the lab materials
 - Handin your code for autograding by the Autolab server
 - View the class scoreboard
 - View the complete history of your code handins, autograded results, instructor's evaluations, and gradebook.
 - View the TA annotations of your code for Style points.

Autolab accounts

- Students enrolled 10am on Mon, Aug 26 have Autolab accounts
- You must be enrolled to get an account
 - Autolab is not tied in to the Hub's rosters
 - If you add in, contact 15-213-staff@cs.cmu.edu for an account
- For those who are waiting to add in, the first lab (datalab) will be available on the Schedule page of the course Web site.

Waitlist questions

- 15-213: Catherine Fichtner (cathyf@cs.cmu.edu)
- 18-213: Zara Collier (zcollier@andrew.cmu.edu)
- 15-513: Catherine Fichtner (cathyf@cs.cmu.edu)

- Please don't contact the instructors with waitlist questions.

Welcome
and Enjoy!