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

这门课和这本书是一个整体，非常紧密的联系在一起

This course,the book and the course are one,they're very tied together

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"!

课程内容参考我们的所见所知

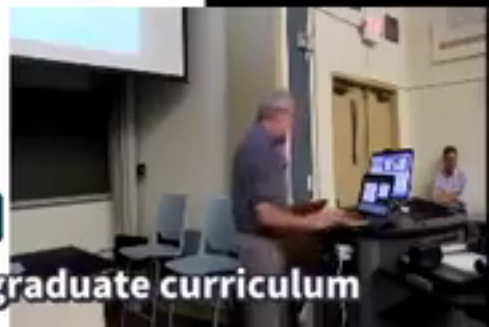
So we refer the the thing we have that

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.

所以这门课有一些内容你们大多数人在本科课程中都曾接触到

So there's a few things about this course for the most part you know in your normal undergraduate curriculum

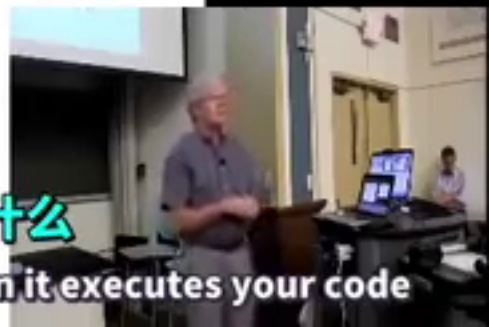


Course Theme:
Abstraction Is Good But Don't Forget Reality

- **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,

这门课的目的是让你深入了解当在执行你的代码时「盒子」在做什么

The purpose of this course is to give you enough understanding of what that box is doing when it executes your code



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.

大规模的软件工程项目、系统硬件设计或计算机技术的任何方面

large-scale software engineering projects,systems hardware design any aspect of computer technology

Course Theme:
Abstraction Is Good But Don't Forget Reality

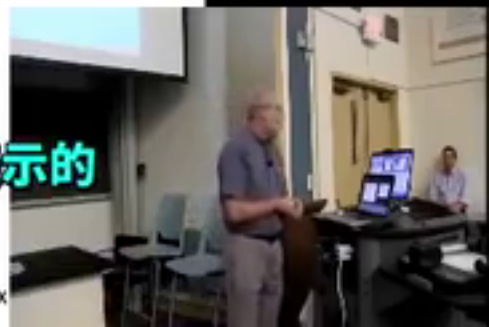
- **Most CS and CE courses emphasize abstraction**
  - Abstract data types
  - Asymptotic analysis
- **These abstractions have limits**
  - Especially in the presence of bugs
  - Need to understand details of underlying implementations
- **Useful outcomes from taking 213**
  - Become more effective programmers
    - 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.

你会从这门课上学到一些概念然后能够学习更局限但是更深层次的运行规律
Where you'll take the sort of ideas from this course and be able to then learn in a somewhat narrower but deeper sense what's really going on

Great Reality #1:
Ints are not Integers, Floats are not Reals

其中一种就是在课程的第一部分中仔细观察数字在计算机中怎么表示的

so one of them is and the first part of the course is going to take a fairly detailed look how numbers are represented in computers

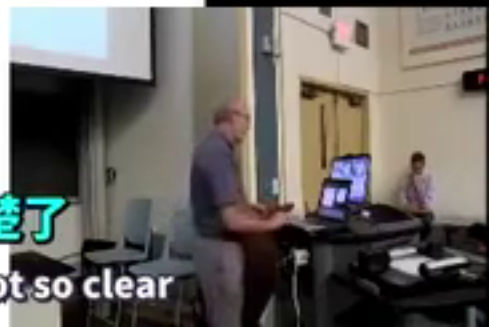


Great Reality #1:
Ints are not Integers, Floats are not Reals

- **Example 1: Is $x^2 \geq 0$?**
  - Float's: Yes!

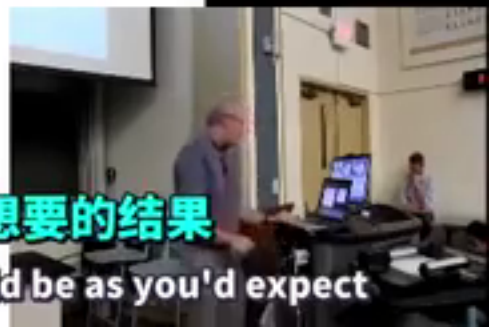
但是对于整数或者 int 类型（计算机中的整数类型）就不是很清楚了

But with integers or int you know the computer representation of integers it's not so clear



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



举个例子，如果你们计算 40,000 的平方，大部分的计算机会给出你们想要的结果

So for example if you square 40,000 on most computers then you'll get whatever that should be as you'd expect

Terminal Shell Edit View Window Help

droh — lldb — 80×36

```
Last login: Mon Aug 24 13:00:02 on ttys001
You have new mail.
macbook> lldb
(lldb) p
```

mako bash ...

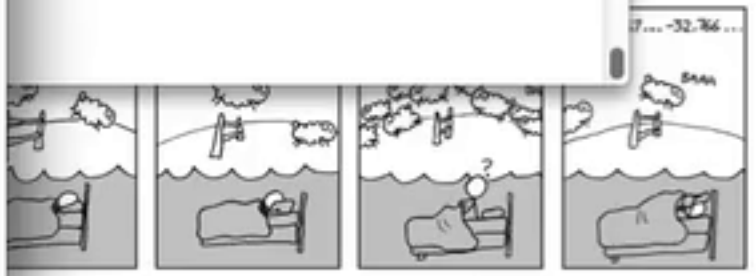
n Presentation

Format Slide Show

Quick Styles Play

Carnegie Mellon

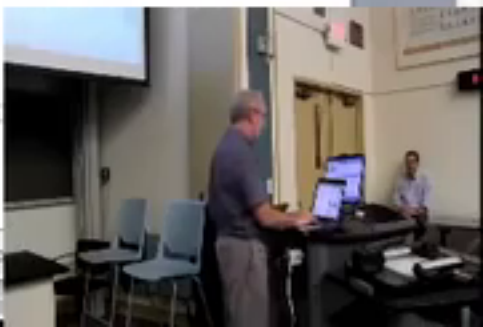
als



00000000

z = x + (y + z)?

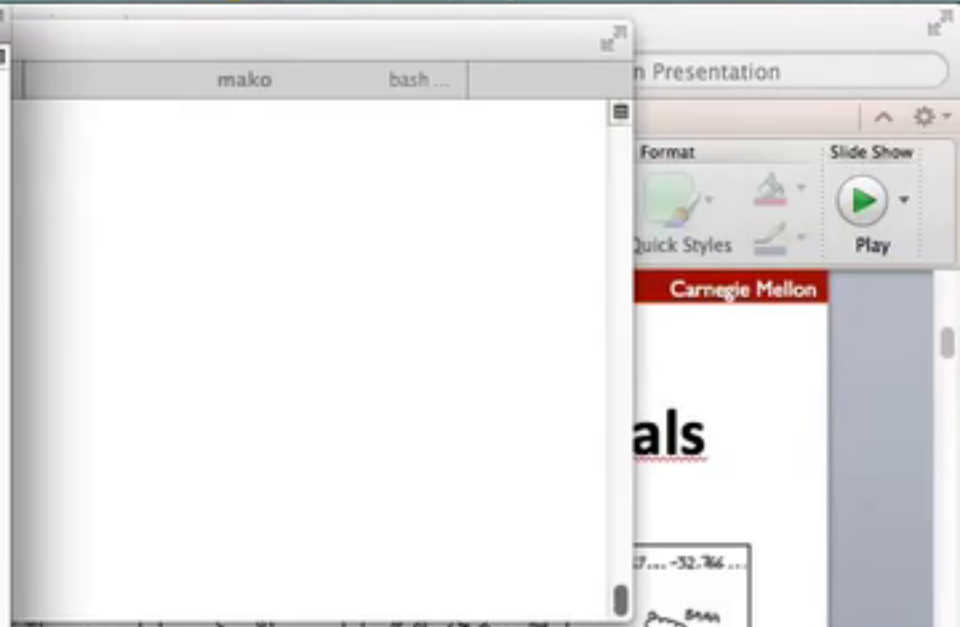
Slide 4 o

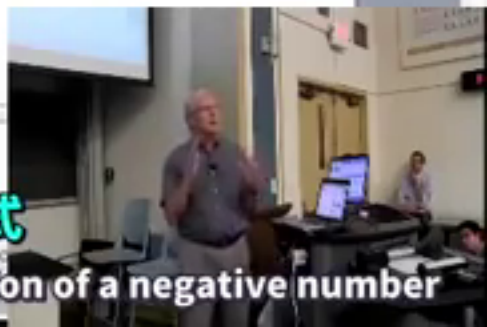


Terminal Shell Edit View Window Help

droh — lldb — 80×36

```
Last login: Mon Aug 24 13:00:02 on ttys001
You have new mail.
macbook> lldb
(lldb) print 40000 * 40000
(int) $0 = 1600000000
(lldb) print 50000 * 50000
(int) $1 = -1794967296
(lldb)
```





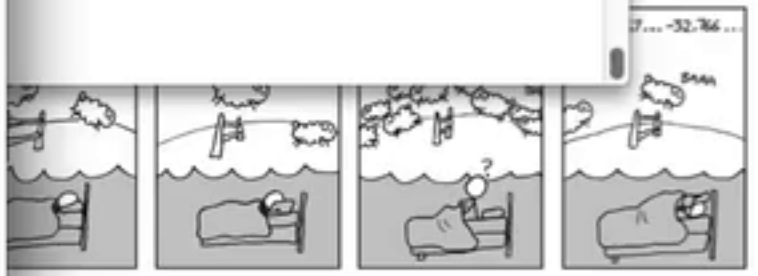
当你做完乘法运算后的得到值的位的格式恰好是一个负数的格式

And the bit pattern that you get when you do this multiplication happens to be the representation of a negative number

Terminal Shell Edit View Window Help

droh — lldb — 80×36

```
Last login: Mon Aug 24 13:00:02 on ttys001
You have new mail.
macbook> lldb
(lldb) print 40000 * 40000
(int) $0 = 1600000000
(lldb) print 50000 * 50000
(int) $1 = -1794967296
(lldb)
```



这是个例子说明你对整数运算的正常期望可能成立，也可能不成立

So that's a an example of where your normal expectations about integer arithmetic may or may not hold

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)?**

下一个问题是它符合加法结合律吗，你能改变顺序后再运算吗?

So then the next question is is a addition associative right can you order the numbers

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 --> 3.14

因为浮点数的取值范围是如此的极端以至于有些数字会消失

Because the range of values you can get in floating pointers so extreme that some numbers kind of disappear on you

Great Reality #1:
Ints are not Integers, Floats are not Reals

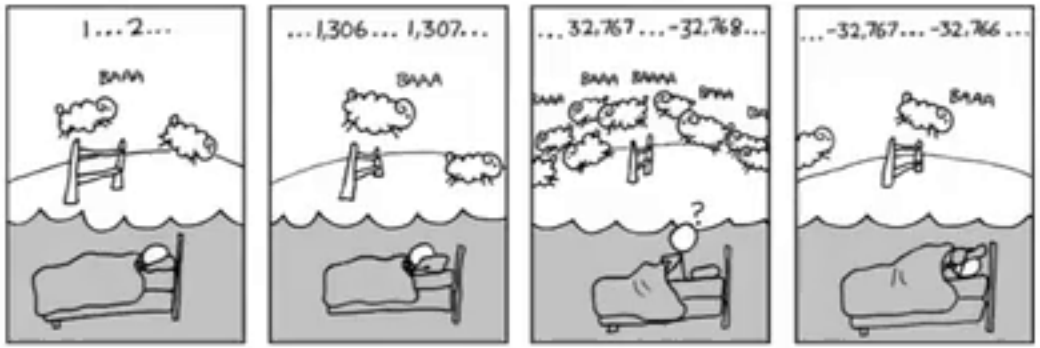
- **Example 1: Is $x^2 \geq 0$?**
  - Float's: Yes!
  - Int's:
    - 40000 * 40000 → 1600000000
    - 50000 * 50000 → ??
- **Example 2: Is (x + y) + z = x + (y + z)?**
  - Unsigned & Signed Int's: Yes!
  - Float's:
    - (1e20 + -1e20) + 3.14 --> 3.14

这一切都归结为一个事实，他们用有限的位组合形式表示在数域中无限扩张的数

And it all comes down to the fact that they use finite representations of things that are potentially infinite in their expanse

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 --> 3.14
    - 1e20 + (-

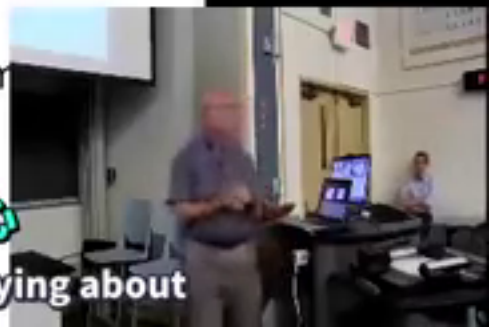
在美国用英语来说至少是一种入睡的方式
Which in the U.S. is in English language at least a way to fall asleep

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 program

我的意思是 90% 的时间，也许你可以通过编写程序而不必担心

I mean 90% of the time maybe you can just get by writing programs and not worrying about



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 program

但它们有时候可能非常重要，例如你正在控制一个火箭

But they sometimes when this could be really important if you're like you know controlling a rocket

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 program

某个聪明人想知道是否可以通过输入了一个负数欺骗这个系统，并让它做坏事

A clever person figured out if I supply a negative number I can fool the system and give it to do bad things

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 program

对于浮点数类似的，如果你要使用浮点数进行严格的计算

Similarly for floating-point if you're going to use floating-point for serious computation

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!

第二步我们将花费大量的时间学习关于机器级编程的意义

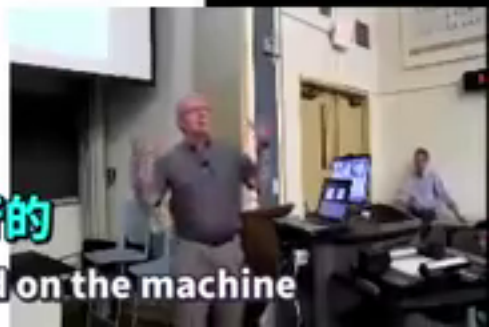
The second is we're going to spend a lot of time in this course learning about machine level programming meaning

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!

你编写的C语言代码是如何变成机器码，以及如何在机器上执行的

How code that you write in C gets turned into machine code and how that gets executed on the machine



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!

一个C编译器，阅读它并理解它，这是一种与你能够自己写汇编所不同的能力

a C compiler and looking at it and understanding it that's a different set of skills than you need to write it on your own

Memory Referencing Bug Example

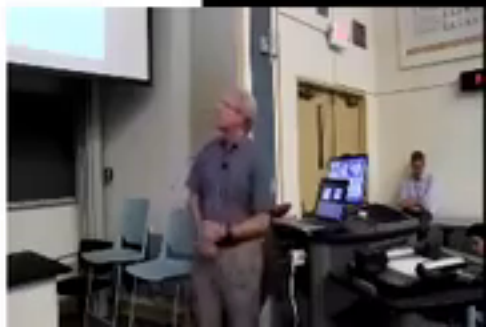
```
typedef struct {
  int a[2];
  double d;
} struct_t;

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
```

- Result is system-dependent

你给这个结构的元素 d 分配了 3.14

That you assigned a 3.14 to element d of this structure

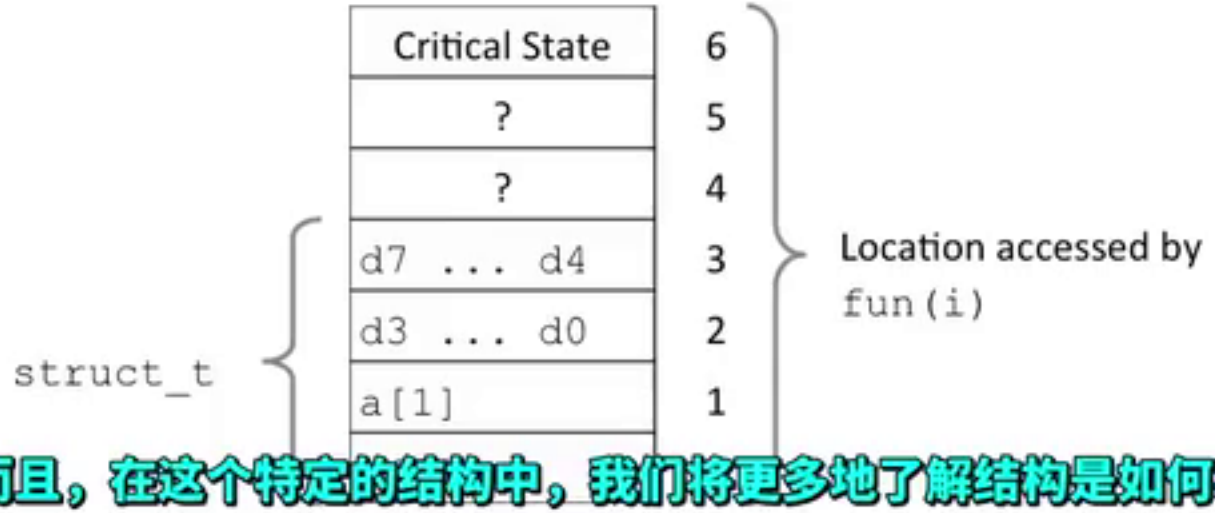


Memory Referencing Bug Example

```
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:



而且，在这个特定的结构中，我们将更多地了解结构是如何实现和排列的

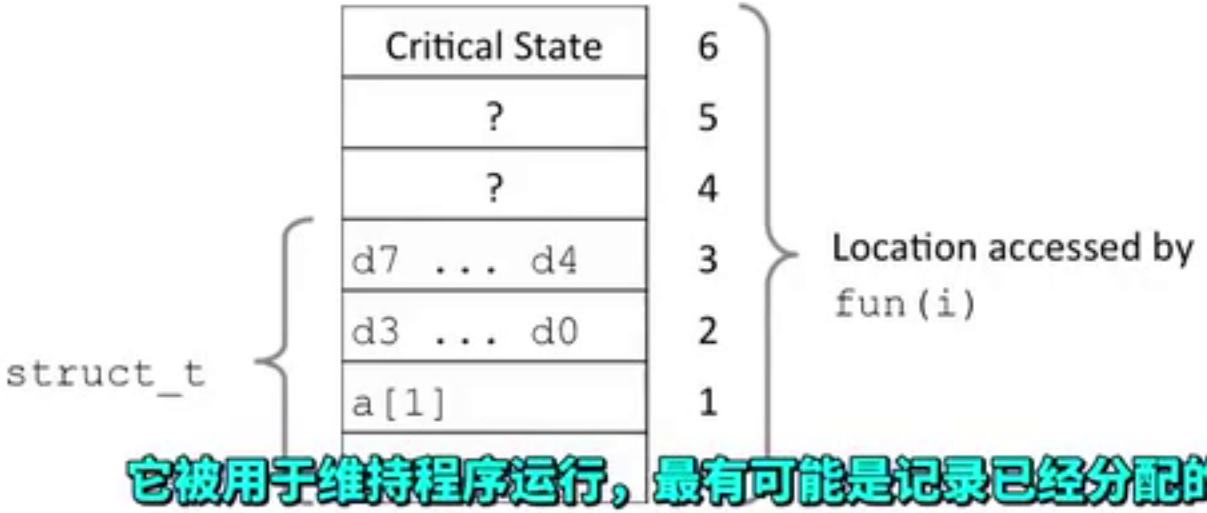
And it in this particular structure and we'll see more about how structures are implemented and weighed out

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:



它被用于维持程序运行，最有可能是记录已经分配的内存

It's using to kind of keep things organized most likely how it keeps track of allocated memory

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

通常情况下，你会导致一些问题，这些问题拥有一种远距离的特性

It's also often the case that you'll cause some problem and it has a sort of action a distance feature that

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

你只是得到更多的经验，你知道有时你应该在你自己的代码中进行边界检查

You just get more experienced and you know at times that you should actually put bounds checking in your own code

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
  - Us[illegible]

因此，理解数据结构的机器级别表示以及它们如何运行对于

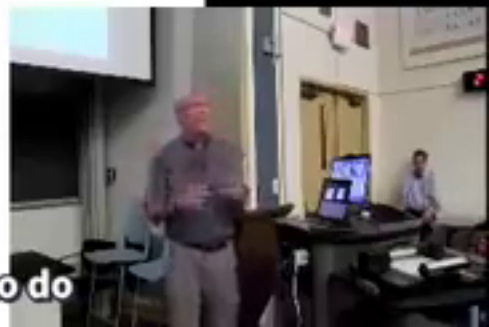
So understanding sort of the machine-level representation of data structures how they work really make a huge difference

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

但是他们之中某些需要进行一些低级别的优化

But they some amount of the sort of low-level optimization that you need to do



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];
}
```

这两个函数在他们的行为方面完全一样

These two functions do exactly the same thing in terms of their...their behavior

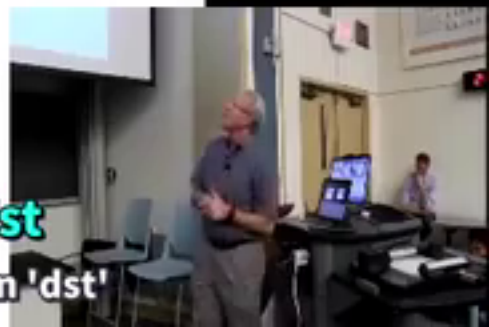
Memory System Performance Example

```
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];
}
```

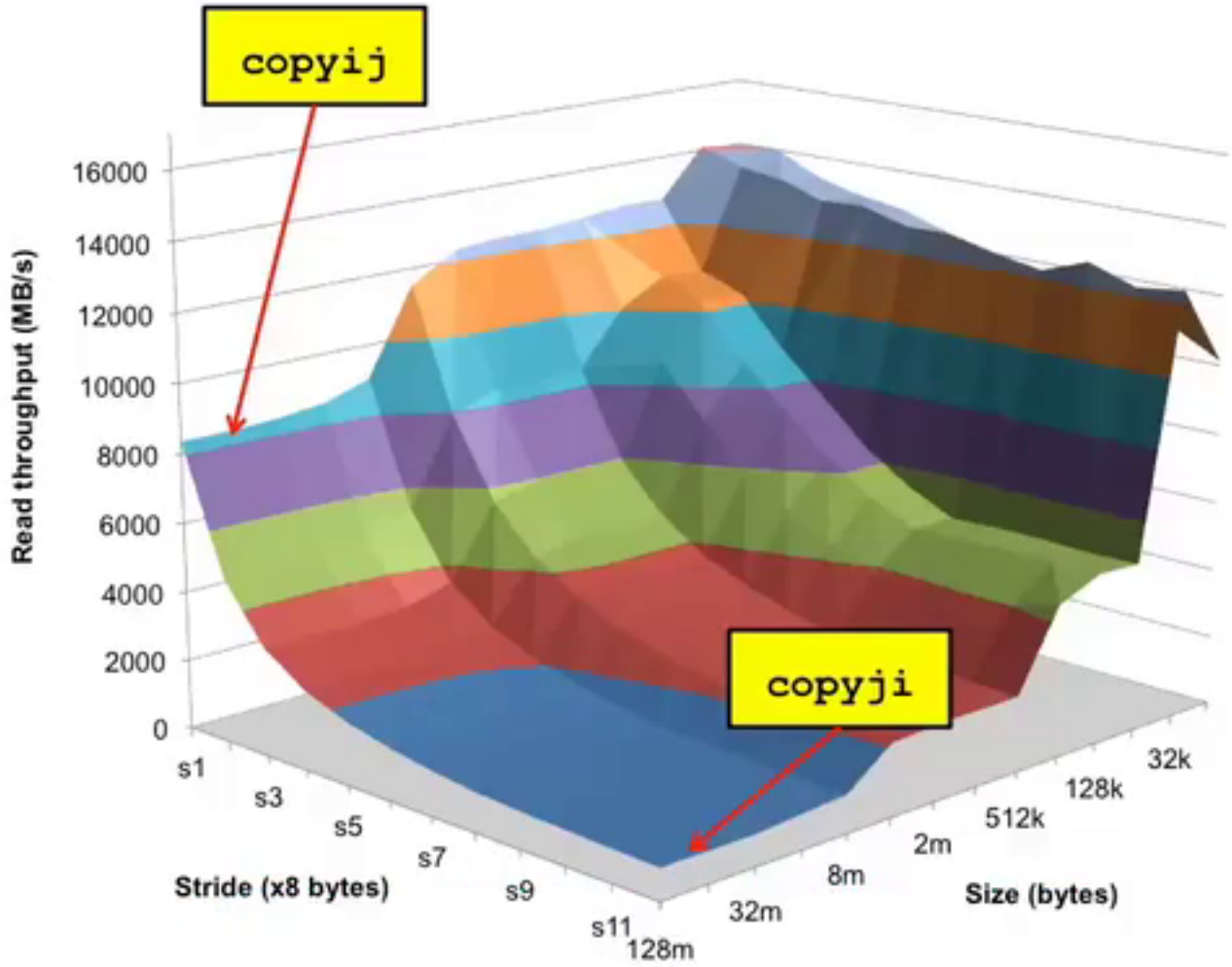
```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

他们所做的是将一个矩阵或数组从源地址 src 复制到目标地址 dst

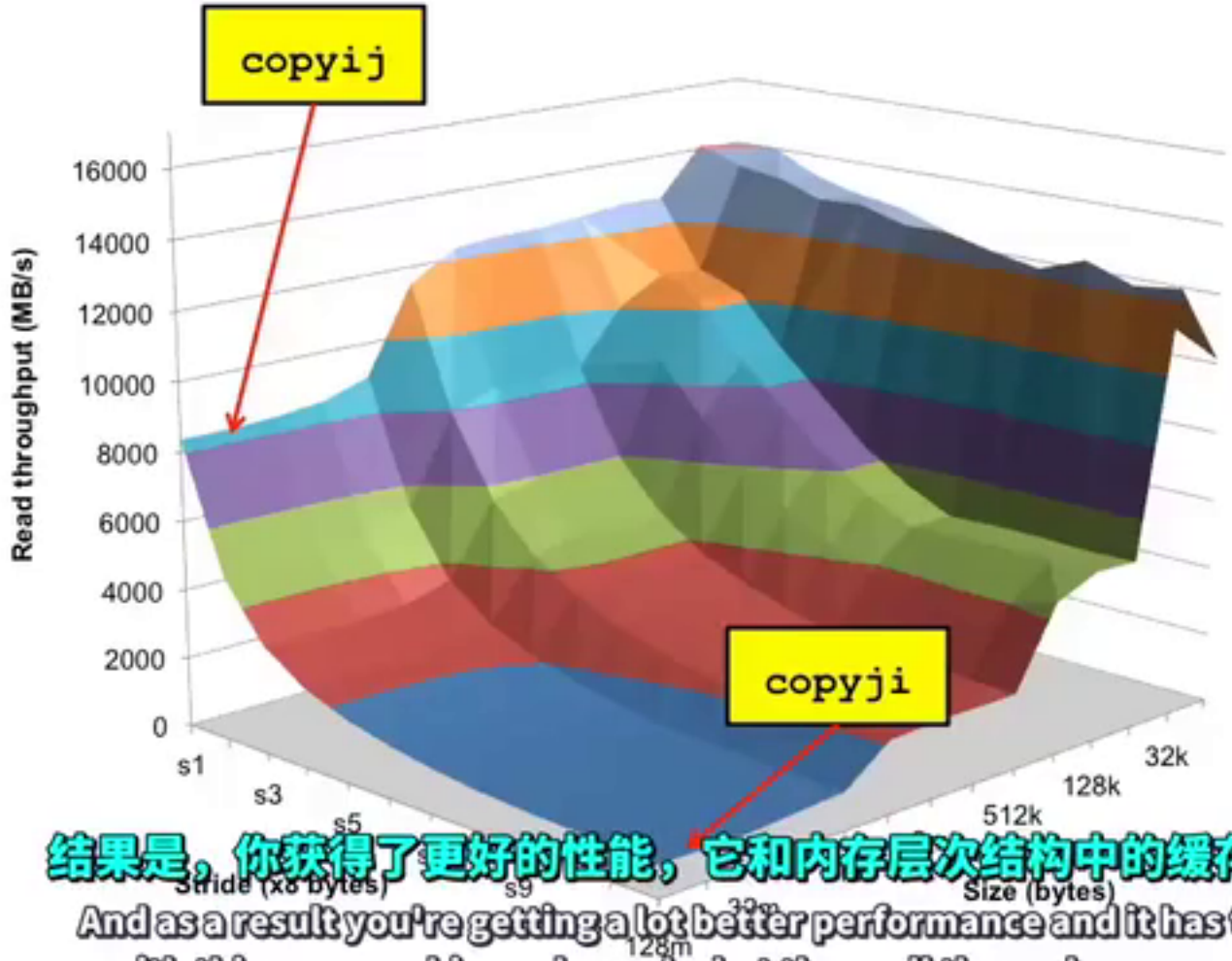
What they do is copy a matrix or array from called source 'src' to a destination 'dst'



Why The Performance Differs



Why The Performance Differs



结果是，你获得了更好的性能，它和内存层次结构中的缓存有关

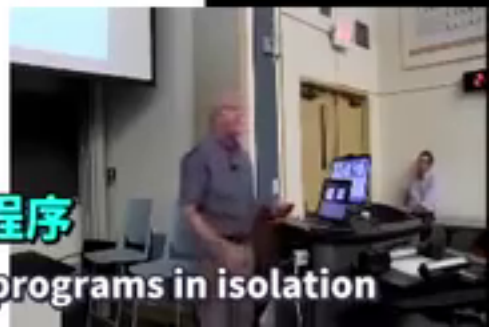
And as a result you're getting a lot better performance and it has to do with this memory hierarchy and what they call the cache memories

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

然后，课程的最后一部分将更多地讨论不仅让计算机孤立运行小程序

And then a final part of the course talks more about not just getting computers to run little programs in isolation

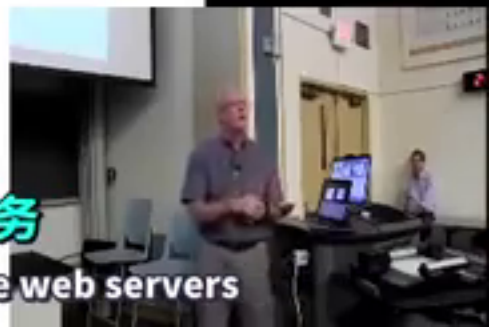


Great Reality #5: Computers do more than execute programs

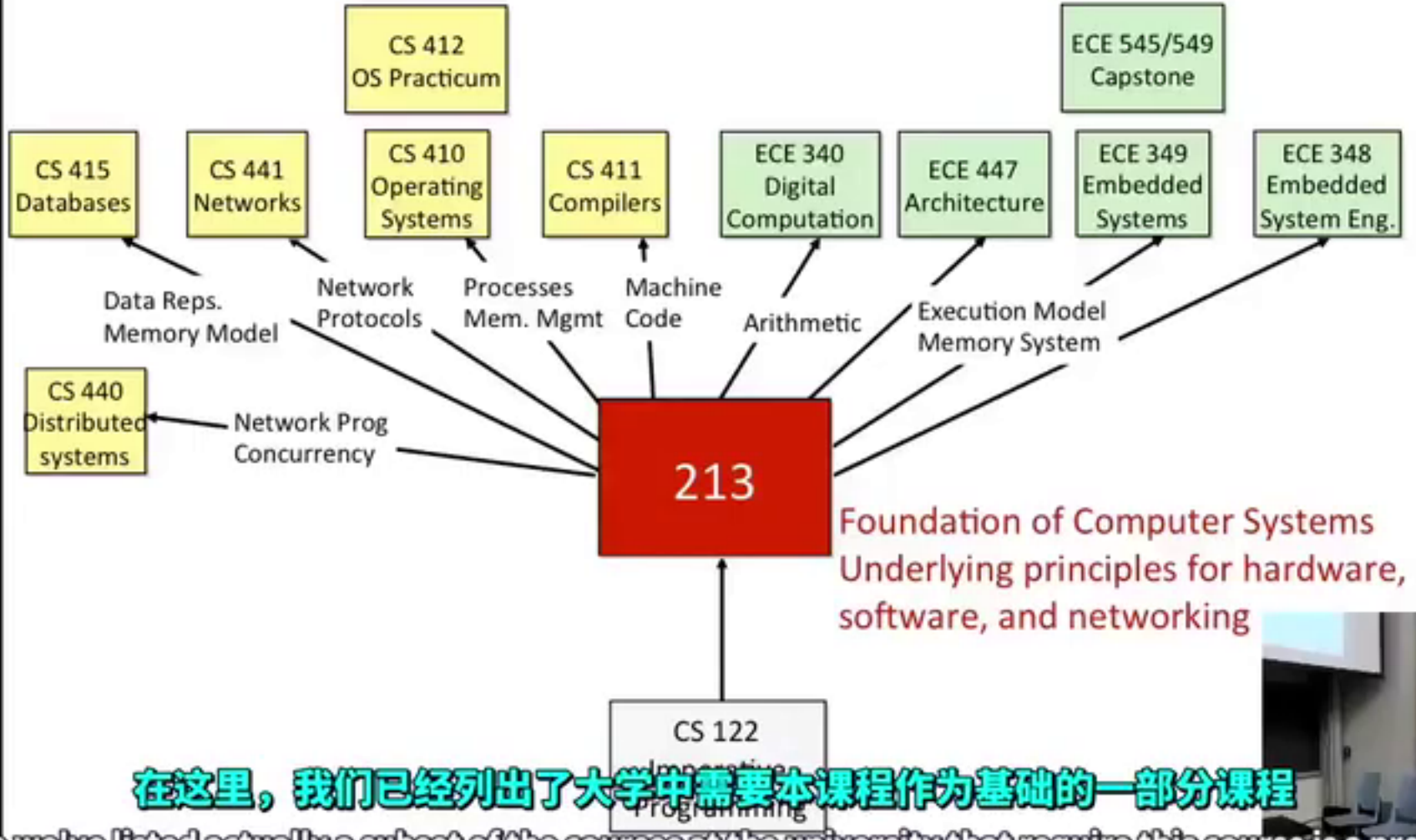
- **They need to get data in and out**
  - I/O system critical to program reliability and performance
- **They communicate with each other over networks**
  - Many system-level issues arise in presence of network
    - Concurrent operations by autonomous processes
    - Coping with unreliable media
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

而且能够通过网络获取彼此交谈，实现像 Web 服务器这样的服务

But getting computers that talk to each other over networks implement services like web servers



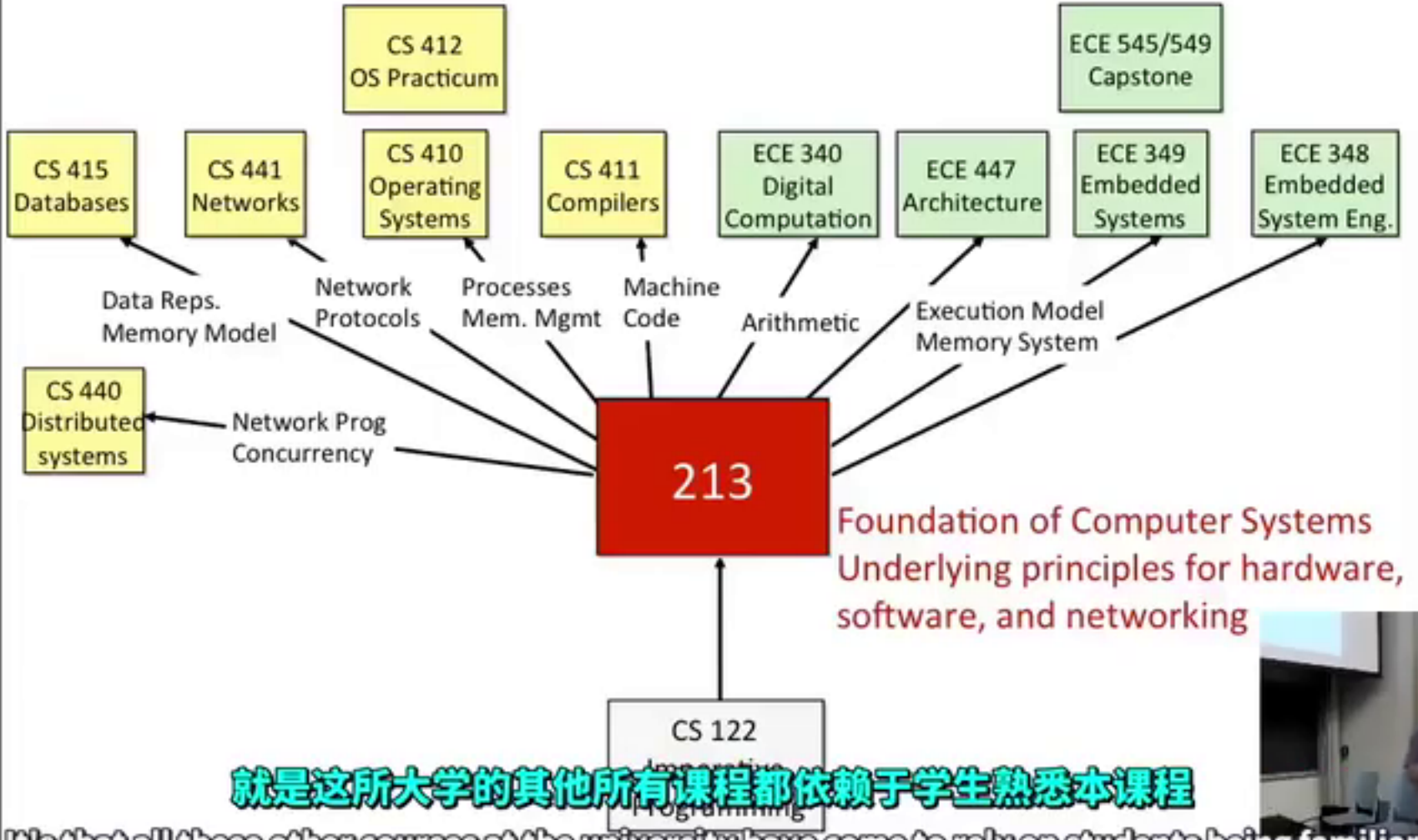
Role within CS/ECE Curriculum



在这里，我们已经列出了大学中需要本课程作为基础的一部分课程

And here we've listed actually a subset of the courses at the university that require this course is a prerequisite

Role within CS/ECE Curriculum



就是这所大学的其他所有课程都依赖于学生熟悉本课程

It's that all these other courses at the university have come to rely on students being familiar

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

大多数系统课程包括你在那里看到的整个课时列表

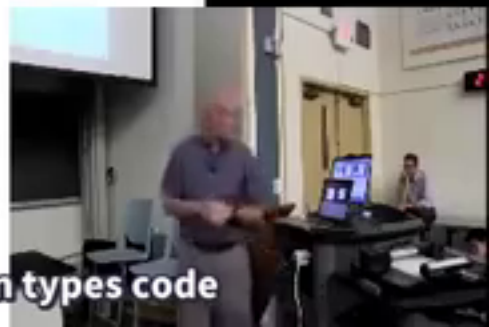
Most systems courses including that whole array you saw there

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

意思是作为一个坐在电脑屏幕前输入代码的程序员去理解

meaning understanding what you as a person who sits in front of a computer screen types code



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 hou
  and staff mailing list.
- All stude并且,o我们会在研究生的课程网页上提供这些录像

And e'll make those available on the course web web page for our graduate students

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 hou
  and staff mailing list.
- All

所以这就是为什么我们有这种累赘的213,513两个版本的原因

So that's the reason why we have this this sort of does not need version of 213, 513

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 hou
  and staff mailing list.

现在 15-213 和 18-213 分别是针对计算机科学和电子计算机工程的本科生

Now 15-213 and 18-213 or are for undergraduates in computer science and ECE respectively

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

我们没有人真的喜欢。但我们必须提起它，那就是学术诚信

None of us really like.But we have to talk about it and that's academic integrity

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

你是如何为你的 Malloc Lab 使用 explict list 或使用 segregated list

You know how are you using an explicit list or using a segregated list for your Malloc lab

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

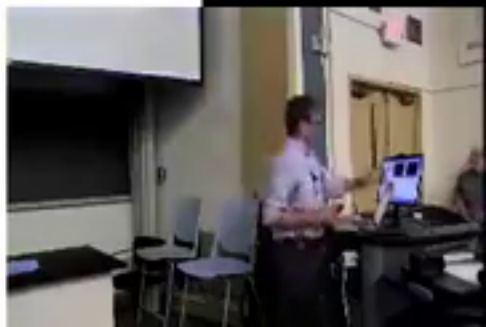
- See the course syllabus for details.
  - Ignorance is not an excuse

你在 Google，Stack Overflow 找你想要的东西，然后剪切并粘贴它

Right you look stuff up on Google, Stack Overflow you cut and paste it

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



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

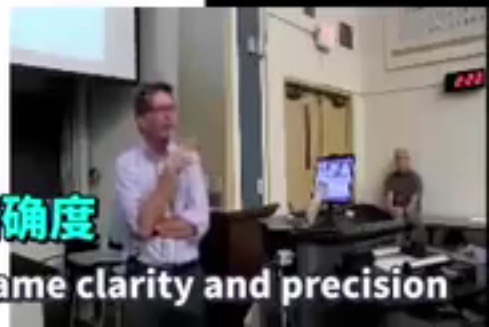
如果你需要更多时间，系统内置有自动扩展时长的插件，以后再说这个，以宽限日为形式

We have but automatic extensions built-in if you need more time I'll talk about that later form of grace days

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



我在写书的时候受到这本书的启发，我试图达到相同的清晰度和精确度

It was one of the inspirations I used when I was writing the book, you know I tried to find that same clarity and precision

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

这门课有四个主要组成部分

There's four main components to the course

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

我们不会改变它，这是固定的，所以你可以看看，并计划你的学期

We don't change it that's it's fixed and so you can look at that and and plan your semester

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

实际上，我们甚至为全世界正在使用本书的其他讲师提前提供了所有的讲座

We've actually even got all of the lectures posted ahead of time for for the other instructors around the world who are using the book

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

由于它是由实验分类的，所以你可以看到常见问题的答案

For it sort of organized by labs,so you can see the answers to frequently asked questions

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

老师们将每天在同一时间和同一地点从 5:45 到 8:30 在 Wayne 5207 大教室

They will be at the same time and same place every day so from 5:45 to 8:30 in WeH 5207 cluster

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

现在大家知道团队工作很重要，而且你将在其他 CS 课程学习如何在团队中工作

Now you know it's important to work in groups and you will learn how to work in groups Other classes CS classes

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

但不是在 213。我们认为这是一节核心课程，我们希望你自己弄清楚

But not in 213 we want this is a kind of a core course we want you to figure stuff out yourself

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

我们所有的作业都使用 Auto Lab 自动实验系统

And all of our hand-ins are using autolab

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

现在在我们的实验室，我们有十台由英特尔捐赠的称为鲨鱼机器的机器

Now for our labs we have ten machines that were donated by Intel called the shark machines

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

然后我们在那之后几年升级，新的那些是咸水鱼机器

And then we upgraded a few years after that and those were the saltwater fish machines

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 run

我们知道的所有这些特殊情况 up

And you know all these special cases what we do

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

现在在演讲厅，你可以使用笔记本电脑

Now in the lecture hall you're permitted to have your laptops

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

现在这个学期的教学大纲

Now rough outline of the semester

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

所以我们检查，当你通过一个阶段时，自动实验室会检查炸弹发来的字符串，

So we check so when you defuse when you defuse a phase the auto lab takes the string that

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

异常条件是我们不给你源代码，我们给你的只是二进制的二进制炸弹

The kicker is we don't give you the source code all we give you is the binary the binary bomb

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

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

你将学习如何编写漏洞，利用返回到一种称为返回到返回导向编程的现代技术

You'll learn how to write exploits using return to a sort of a modern technique called return to return oriented programming

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

而且禁止，也不可能在栈上执行代码的事实

And prohibitive and make it impossible to execute code on the on the stack

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.

这将涉及到对内存层次结构如何工作的以及如何利用它的理解

And this will involve sort of understanding of how the memory hierarchy works and how to exploit it

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

这个想法是，你了解了一种特殊的控制流程，如低级别的硬件中断和异常

And so the idea is that you cover an exceptional control flow like low hard, low-level hardware interrupts and exceptions

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

所以，这就是你无论何时登录到 Linux 机器时，与其交互就要用到的命令行程序

So that's the program the command line program that you interact with whenever you login a Linux box

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.

这是互联网与世界上任何可交互的机器交流的基本界面

Which is the basic interface for the internet to talk to machines any potentially any machine in the world

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.