

# Course Overview

Introduction to Computer Systems  
1<sup>st</sup> Lecture, Sep 9, 2024

## Instructors

**Class 1: Chen Xiangqun**

**Liu Xianhua**

**Class 2: Guan Xuetao**

**Class 3: Lu Junlin**

# 课程概述

计算机系统导论  
第一讲，2024年9月9日

## 教师


**1班： 陈向群**

**刘先华**

**2班： 管雪涛**

**3班： 陆俊林**

# 主要内容

- 课程起源 
- 课程规划
- 五个有趣的现实问题
- 课程主体内容
- 注意事项

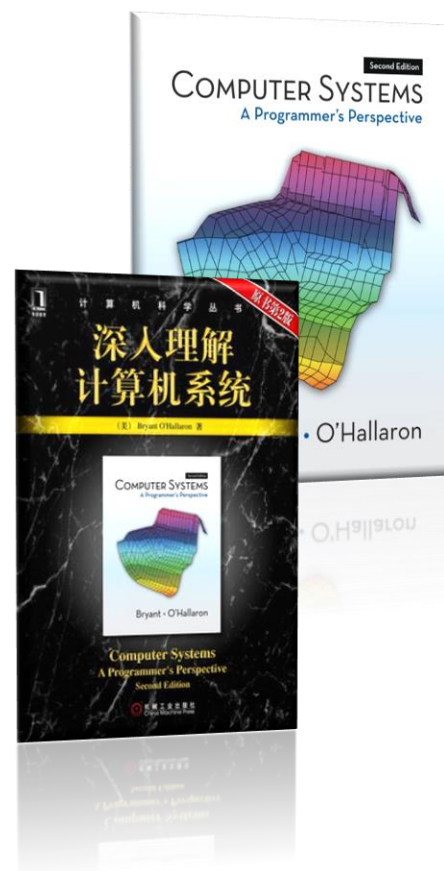
# 课程起源

## ■ 创立：

- 卡耐基梅隆大学计算机科学学院创立
- 全球超过180所大学采用了该课程教材、设立了相同或类似的课程
- 特点：注重实践（程序员视角）、强调对系统的理解

## ■ 发展

- 2012年，北大信息科学技术学院与卡耐基梅隆大学计算机科学学院联合对该课程进行升级，并正式引入国内



# 合作建设课程



**课程特点：  
影响面广，关注度高**



# 北京大学的课程规模

## ■ 2010-2011 学年，本科班级规模的初步统计

- 20 人以下的班级占有所有本科课程的比例仅为3.8%，100 人以上的课程约占27.2%（进一步统计表明，200 人以上的班级占4%）
- 在全校153 个20 人以下的小班中，大部分是外国语学院课程，大约占67%，其他为公共英语课程
- 这表明，当时在绝大多数院系中小班教学很少开展

	20人以下	20~39人	40~49人	50~99人	100人以上	合计
课程数	153	1387	186	1205	1093	4024
百分比	3.8%	34.5%	4.6%	29.9%	27.2%	100%

# 北京大学本科生“研讨型小班教学”试点


## ■ 2012年秋开展第一批试点（已经连续开课10年）

- 五个学院，六门必修基础课

信息科学技术学院	“计算机系统导论”
数学科学学院	“数学分析”、“抽象代数”
物理学院	“量子力学”
化学与分子工程学院	“无机化学”
生命科学学院	“生物化学”

- 2013年春，信息学院增设小班课“算法分析与设计”

# 主要内容

- 课程起源
- 课程规划 
- 五个有趣的现实问题
- 课程主体内容
- 注意事项

# 本课程的教学方式

## ■ 研讨型教学的两种主要方式

- 第一种，一学期由一个教师面对一个小班的学生
- **第二种，大班讲授课教学同时辅以小班研讨课**

## ■ 本课程采用上面第二种方式

- 每周两次，大班授课
- 每周一次，小班研讨



# 课程的时间、地点和特殊情况说明

## ■ 地点安排

- 1班：二教207；2班：一教201；3班：理教311

## ■ 时间安排

- 周一5~6节和周四5~6节，大班
- 周三10~11节，小班

## ■ 特殊情况说明

- 9.14（周六）：安排9.16（周一）的课，9.16放假
- 9.30（周一）：正常上课
- 10.2（周三）、10.3（周四）、10.7（周一）：放假
- 11.4（周一）：期中考试
- 11.7（周四）：专题讲座（三个班合上）

# 大班课程安排

## ■ 上半学期

- 大致覆盖教材第一部分（Part I），即第2~6章
- 2. 信息的表示和处理
- 3. 程序的机器级表示
- 4. 处理器体系结构
- 5. 优化程序性能
- 6. 存储器层级结构

# 大班课程安排

## ■ 下半学期

- 大致覆盖教材第二、三部分（Part II/III），即第7~12章
- 7. 链接
- 8. 异常控制流
- 9. 虚拟内存
- 10. 系统级I/O
- 11. 网络编程
- 12. 并发编程



**课程特点:**  
**课时多, 教学内容多**





## 课程特点： 大班教学和小班研讨结合



# 实验题系统

**课程特点：**  
**学生在指定系统上完成实验题**

## ■ 大型特色实验题


- 从实际问题出发
- 具有很强的趣味性
- 平均每两周完成一个



## ■ 实验题智能评价系统

- 自动根据性能、时间、提交次数等对学生提交的实验题进行评分
- 实时公开发布所有同学完成情况并分步分题进行比对，鼓励学生对实验的钻研

# 主要内容

- 课程起源
- 课程规划
- 五个有趣的现实问题 
- 课程主体内容
- 注意事项

# 本课程关注的问题和目标

## ■ 本课程关注的问题：

- 计算机抽象概念与实际计算机系统之间的差异
- 计算机抽象概念在实际计算机系统上的实现方式

## ■ 本课程的目标：

- 为初入计算机专业的学生建立计算机系统的整体知识框架
- 训练学生养成良好的编程习惯，进而具备更为高效的编程能力，尤其是提高程序的性能、可移植性和健壮性等方面
- 为学生后续学习编译、网络、操作系统、计算机体系结构等专业课程奠定基础



# 本课程独特的视角

- 本课程是**从编程者角度出发**，描述计算机系统如何执行程序、存储信息和通信
- **涵盖计算机系统**从上到下的多个层次，包括：
  - 机器语言及其如何通过编译器优化生成
  - 程序性能评估和优化
  - 存储结构组织和管理
  - 网络技术和协议
  - 并行计算的相关知识

# 问题1：整型不是整数，浮点型不是实数

## Ints are not Integers, Floats are not Reals

### ■ 例1.1: $x^2 \geq 0$ 永远成立吗?

- 如果  $x$  是浮点型，成立
- 如果  $x$  是整型
  - $40000 * 40000 \rightarrow 1600000000$
  - $50000 * 50000 \rightarrow$  **负数**，因为整型有上界溢出

### ■ 例1.2: 是否满足结合律 $(x + y) + z = x + (y + z)$ ?

- 如果  $x, y, z$  是整型，满足结合律
- 如果  $x, y, z$  是浮点型
  - $(1e20 + -1e20) + 3.14 \rightarrow 3.14$
  - $1e20 + (-1e20 + 3.14) \rightarrow$  **0**，因为浮点数精度不同不满足结合律

# 计算机系统算术 $\neq$ 数学中的算术(1/2)

## ■ 整数性质

- 交换律:  $a+b = b+a$
- 结合律:  $(a+b)+c=a+(b+c)$
- 分配律:  $a \cdot (b+c)=a \cdot b + a \cdot c$
- 整型运算满足以上性质

## ■ 实数性质

- 单调性: if  $a \geq b, c \geq 0$ , then  $(a+c) \geq (b+c)$
- 浮点型运算满足单调性

# 计算机系统中的算术 $\neq$ 数学中的算术(2/2)

## ■ 有些性质在计算机系统中并不成立

- 计算机系统只能表示“**有限大小的数**”：溢出问题（例1.1）
- 浮点型不满足结合律：**舍入操作会造成精度误差**（例1.2）
- 需要记住计算机中不同数据类型所满足的数学性质
- 对编译器和科学计算程序员尤为重要：因为缺少一些数学性质会使得解决某些简单问题变得麻烦。

## ■ 例1.3:

- 两个整型a和b是否相等：  $a == b$  😊
- 两个浮点型a和b是否相等：  $a == b$  😞
  - 因为两个数精度可能不同
  - 正确方法——作差取绝对值

$\text{fabs}(a-b) \leq \text{epsilon}$ , (epsilon是很小的数, 如0.00001)

## 问题2：了解汇编 (1/4)

### You've Got to Know Assembly

可能你永远都不会去写汇编程序，但是.....

有助于了解机器层面的程序执行模型

#### ■ 帮助查找底层实现相关的程序错误 (bug)

- 例2.1：比较整型(int)、无符号整型(unsigned int)
- $d = -1 < \text{TOTAL} = 12$ ，理应输出small，但结果却是large
  - sizeof()的返回值是unsigned int;
  - if语句作比较时，编译器认为-1是unsigned int (很大的整数)
- 通过底层汇编代码/目标程序文件(二进制文件)查看 d 的数值

```
int array[] = {1,2,3};  
#define TOTAL sizeof(array) /* unsigned int */  
void main() {  
    int d = -1;  
    if (d <= TOTAL)  
        printf("small\n");  
    else printf("large\n");  
}
```

# 问题2：了解汇编 (2/4)

## You've Got to Know Assembly

### ■ 程序性能调优

- **例2.2：**尝试不同代码写法，分析比较不同的底层汇编代码效率
- 两个程序似乎有相同的行为。但是fun2的效率会更高
- 通过底层代码可以看出，fun1需要6次存储器引用，而fun2只需3次

```
void fun1(int *x, int *y)
{
    *x += *y;
    *x += *y;
}
```

```
void fun2(int *x, int *y)
{
    *x += 2* (*y);
}
```

## 问题2：了解汇编 (3/4)

### You've Got to Know Assembly

#### ■ 系统软件或嵌入式软件开发

- 例如系统软件工程师往往会要求写小段汇编代码
- **例2.3**：把小段汇编代码加入C代码，来访问硬件（处理器）上的周期计数器（cycle counter）。

```
static unsigned cyc_hi = 0;
static unsigned cyc_lo = 0;

/* Set *hi and *lo to the high and low order bits
   of the cycle counter.
*/
void access_counter(unsigned *hi, unsigned *lo)
{
    asm("rdtsc; movl %%edx,%0; movl %%eax,%1"
        : "=r" (*hi), "=r" (*lo)
        :
        : "%edx", "%eax");
}
```

## 问题2：了解汇编 (4/4)

### You've Got to Know Assembly

- 防范恶意软件或分析第三方软件的安全性
  - 分析没有源代码的软件时，需要进行**反汇编**
  - 常见的安全漏洞包括：缓冲区溢出、内存泄露、非授权内存写入等
  - 对反汇编得到的代码进行**静态分析**，是一种找到已知安全漏洞代码的有效手段
  - **例2.4**：定位 **gets()** 这样不安全函数对应的汇编代码

```
void main{}  
{  
    char buf[1024];  
    gets(buf);  
    /*用户输入不做限制，缓冲区溢出*/  
}
```

```
#define BUFSIZE 1024  
void main{}  
{  
    char buf[BUFSIZE];  
    fgets(buf, BUFSIZE, stdin);  
    /*限制输入大小的参数*/  
}
```



# 问题3：内存对程序性能的影响至关重要

## Memory Matters

## Random Access Memory Is an Unphysical Abstraction

### ■ 内存是有限的

- 必须合理地分配和管理内存
- 很多程序受限于内存

### ■ 内存引用错误尤为严重

- 错误的危害因时间、空间而异

### ■ 内存性能并不是始终如一的

- 高速缓存和虚拟内存极大地影响程序性能
- 根据存储系统的特点，可以对程序进行调优(见问题4)

# 内存引用错误 (1/3)

```
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)	→	3.1399998664856
fun(3)	→	2.00000061035156
fun(4)	→	3.14
fun(6)	→	segmentation fault

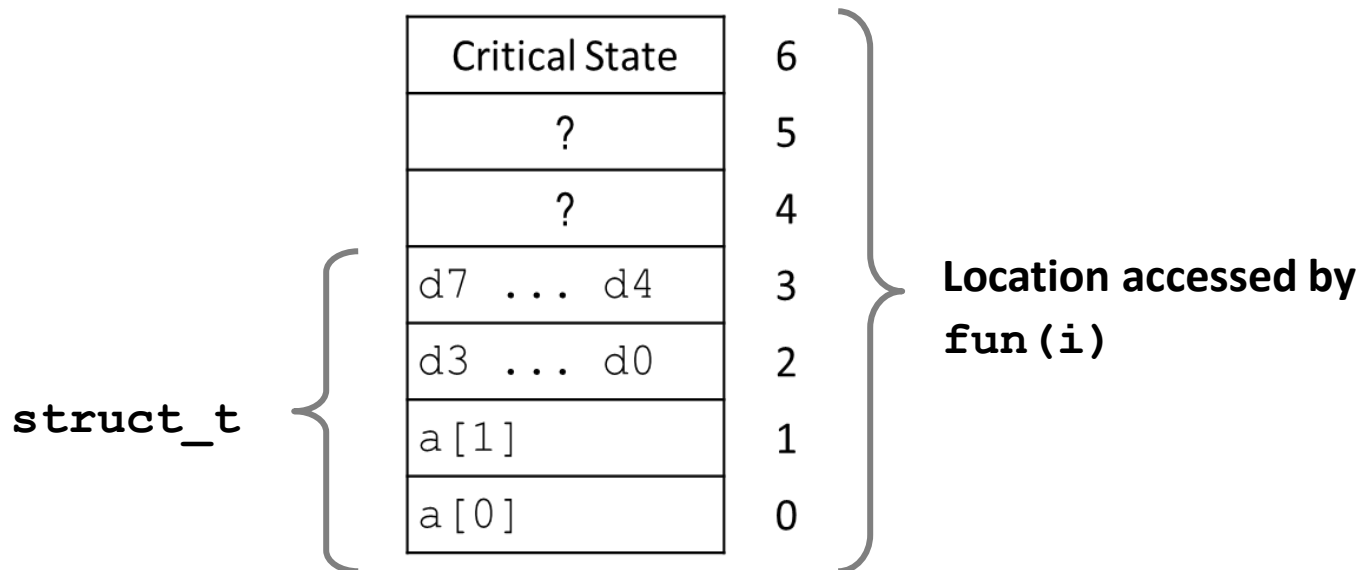
- Result is system specific

# 内存引用错误 (2/3)

```
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:



# 内存引用错误 (3/3)

- C 和 C++ 并没有提供对此类错误的防范机制，比如：
  - 数组越界错误
  - 指针错误
  - 滥用 malloc/free 函数
- 应对措施
  - 用其他语言编程，例如 Java, Ruby, Python, ML
  - 使用工具来检测此类内存错误

## 问题4：算法性能分析结果 $\neq$ 实际程序性能

### There's more to performance than asymptotic complexity

- 代码写的好坏与否，可能导致程序性能的数量级差别
- 程序性能优化有多个层面
  - 算法，数据表达，过程，循环
- 只有理解了系统实现才能做到有效优化
  - 衡量程序性能的指标：执行时间、内存占用、能耗等。
  - 了解程序的编译、执行过程中的细节，如内存访问模式
    - 例：内存访问模式影响程序性能

# 内存性能影响程序性能

```
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];  
}
```

**4.3ms**

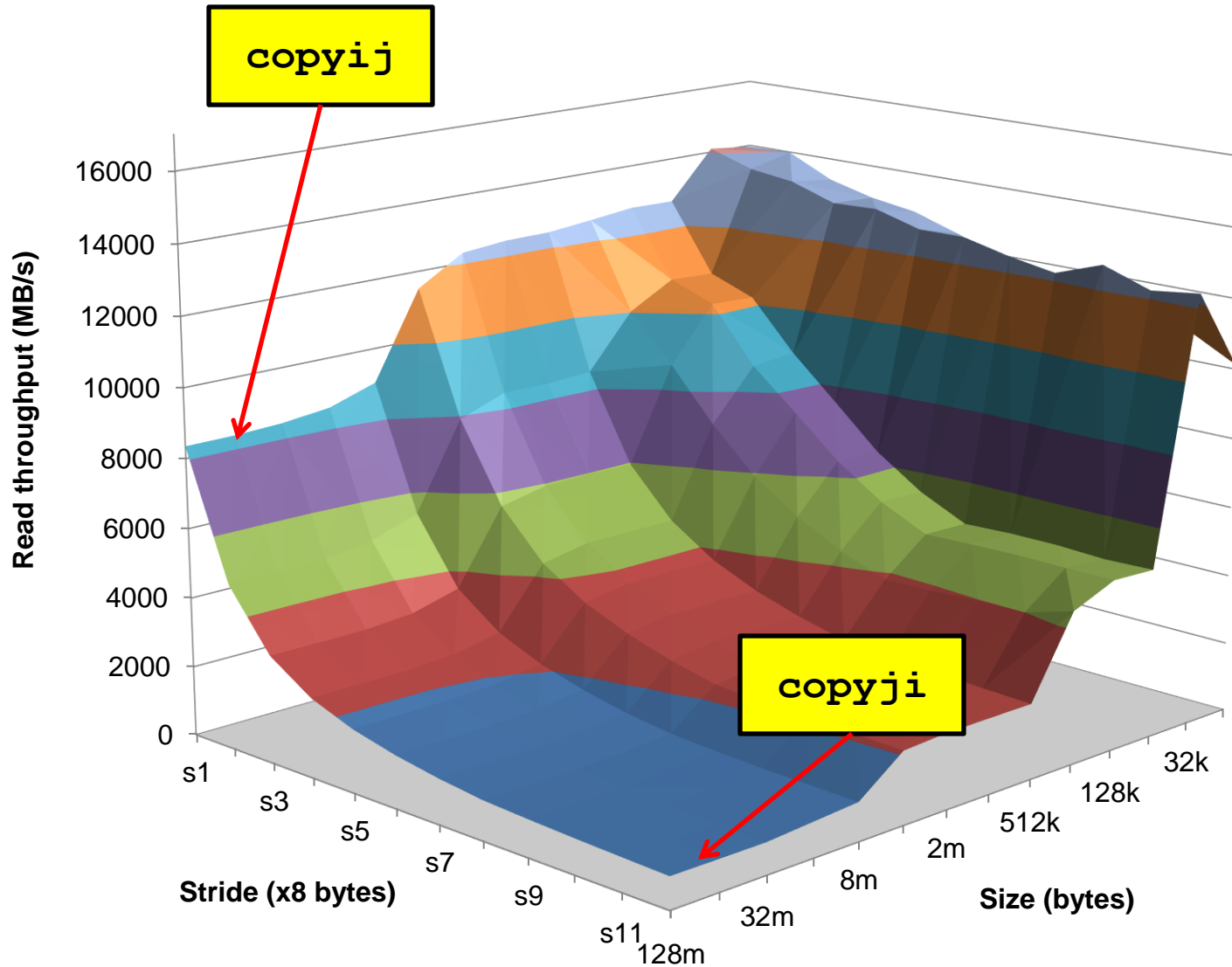
```
void copyji (int src[2048][2048],  
             int dst[2048][2048])  
{  
    int i,j;  
    for (j = 0; j < 2048; j++)  
        for (i = 0; i < 2048; i++)  
            dst[i][j] = src[i][j];  
}
```

**81.8ms**

2.0 GHz Intel Core i7 Haswell

- 内存是分层组织的
- 程序性能取决于内存访问模式
  - 例如：如何访问内存中的二维数组、多维数组

# 为什么性能有这些差别



# 问题5：计算机网络环境下的新问题

## Computers do more than execute programs

### ■ 计算机需要输入和输出数据

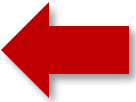
- 程序执行前，需要输入数据
- 程序执行后，需要输出结果
- 在**网络环境**下，数据输入来源
  - 本地磁盘
  - 网络中别的计算机。例如，利用上传数据到服务器，利用服务器的超强计算能力做仿真实验

### ■ I/O 系统对程序稳定性和性能至关重要

- 如果缺少I/O异常处理能力，就会出现程序运行错误



# 主要内容

- 课程起源
- 课程规划
- 五个有趣的现实问题
- 课程主体内容 
- 注意事项

# 课程主体内容

- ① 程序与数据
- ② 处理器体系结构
- ③ 程序性能
- ④ 分级存储器体系
- ⑤ 异常控制流
- ⑥ 虚拟内存
- ⑦ 网络、并发

**Programs and Data**

**Processor Architecture**

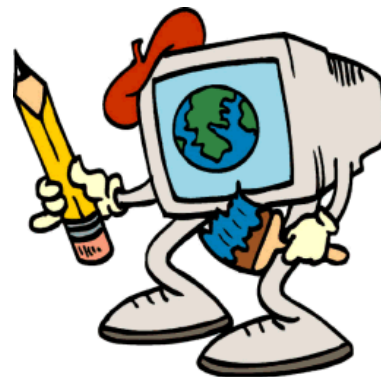
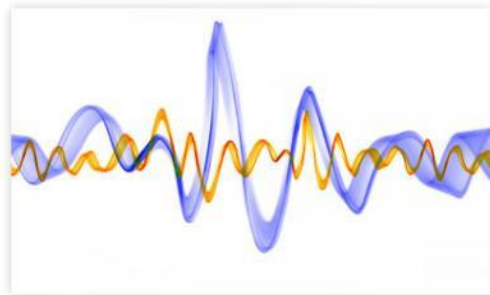
**Performance**

**The Memory Hierarchy**

**Exceptional Control Flow**

**Virtual Memory**

**Networking, and Concurrency**



# 一、程序与数据

## Programs and Data (1/2)

### ■ 主要知识点

- bits operations, arithmetic, assembly language programs
- representation of C control and data structures

### ■ 涉及领域

- architecture and compilers

### ■ 对应的实验题 (Labs)

- L1 (data lab): Manipulating bits. **位级数据操作实验**

在严格限制操作类型的前提下，通过编程解决一系列“难题”，实现各种“简单”的逻辑和算术功能。

该实验可以帮助学生理解各种数据类型的位一级的表达方式，以及位一级的数据操作的实际行为，同时，加深学生对二进制和硬件指令的理解。

# 一、程序与数据

## Programs and Data (2/2)

- L2 (bomb lab): Defusing a binary bomb. **拆解二进制炸弹实验**

“二进制炸弹”是一个趣称，该程序为二进制可执行程序；拆解过程共分为6个关卡，需要学生分别输入6次正确的数据来进行拆解，如果任何一次数据错误，则会引爆炸弹，导致拆解失败。

该实验可以帮助学生理解高级语言是如何编译成汇编语言的，并且，在实验过程中，加深了学生对于数据在内存中的存储方式，以及数据、指针和指令等各种知识点之间的关联和理解。

- L3 (attack lab): Hacking a buffer bomb. **缓冲区溢出实验**

缓冲区溢出是操作系统和网络服务器的一种常见安全隐患。本实验通过模拟缓冲区溢出攻击，达到修改程序运行时行为的目的，来帮助学生理解栈的组织方式和重要性，以及缓冲区溢出的本质原理，同时增强学生对于计算机系统的安全防范意识。

# 二、处理器体系结构 和 程序性能

## Processor Architecture & Performance

### ■ 主要知识点

- Instruction Set Architecture
- sequential and pipeline processors
- co-optimization (control and data)
- measuring time on a computer

### ■ 涉及领域

- architecture, compilers, and OS

### ■ 对应的实验题 (Labs)

- L4 (arch lab): optimizing a pipelined processor and a benchmark program **处理器结构实验**

本实验需要优化一个流水线处理器和一个评测程序，使得程序在处理器上运行时的性能尽可能好。

该实验帮助学生更好地理解处理器的体系结构，并通过程序实践使得学生能够更好地掌握和应用优化程序性能的各种方法。

# 三、分级存储器体系

## The Memory Hierarchy

### ■ 主要知识点

- memory technology
- memory hierarchy
- caches, disks, locality

### ■ 涉及领域

- architecture and OS

### ■ 对应的实验题 (Labs)

- L5 (cache lab): Building a cache simulator and optimizing for locality.

#### 性能优化实验

本实验需要优化两个矩阵算法的变换和计算，以获得尽可能好的应用程序性能。

本实验帮助学生更好地理解高速缓存的特性和重要性，并通过程序实践增强学生对于底层程序优化的认识。

# 四、异常控制流

## Exceptional Control Flow

### ■ 主要知识点

- hardware exceptions, processes, process control
- Unix signals, nonlocal jumps

### ■ 涉及领域

- compilers, OS, and architecture

### ■ 对应的实验题 (Labs)

- L6 (tsh lab): Writing your own Unix shell. **定制shell程序实验**

本实验需要实现一个简单的shell程序，该程序需要包括作业控制，如ctrl-c和ctrl-z等按键的处理，前台、后台等方式的实现。

本实验帮助学生理解应用程序级别如何实现并行，并通过程序实践增强学生对于进程控制、信号、信号处理等内容的认识。

# 五、虚拟内存

## Virtual Memory

### ■ 主要知识点

- virtual memory
- address translation
- dynamic storage allocation

### ■ 涉及领域

- architecture and OS

### ■ 对应的实验题 (Labs)

- L7 (malloc lab): Writing malloc package. **动态内存管理实验**

本实验需要实现一个动态内存管理器，包括malloc、free和realloc接口函数。

该实验帮助学生理解数据布局和组织，并要求学生权衡不同实现方案的空间和时间的性能。



# 六、网络和并发

## Networking, and Concurrency

### ■ 主要知识点

- high level and low-level I/O, network programming
- Internet services, Web servers
- concurrency, concurrent server design, threads
- I/O multiplexing with select

### ■ 涉及领域

- networking, OS, and architecture

### ■ 对应的实验题 (Labs)

- L8 (proxylab): Writing your own Web proxy. **Web代理实验**

本实验需要实现一个Web代理服务，即当Web浏览器希望访问Web服务器上的页面时，其实际上是从代理服务器上获取数据。

该实验帮助学生理解网络程序设计和HTTP协议的基本原理，并让学生在实践中理解和掌握并发和同步这两个关键的基本概念。

# 实验题 (LAB)

L1	Datalab	位级数据操作实验
L2	Bomblab	拆解二进制炸弹实验
L3	Attacklab	缓冲区溢出实验
L4	Archlab	处理器结构实验
L5	Cachelab	性能优化实验
L6	Tshlab	定制shell程序实验
L7	Malloclab	动态内存管理实验
L8	Proxylab	Web代理实验


## ■ 说明:

- 使用Autolab下载和提交, 注意提交版本数量有限制 (通常16个)
- 发布时间通常为对应的大班课后
- 发布后到规定日期的23:59为due time, 2天后23:59为deadline
- Grace day总共为5天, 仅用于前7个LAB
- 具体要求以每个LAB的发布说明为准
- 第一次小班课上会讲解LAB相关的基本操作

# 每个实验必须独立完成

- 每次LAB都有可能抽查代码重合度，对比对象包括本次作业、往年作业和网上代码
- 如果发现抄袭，根据严重程度，可能的后果包括但不限于：
  - 本次LAB 0分
  - 全部LAB 0分
  - 本课程不及格
- 提供代码的学生同样处罚

# 主要内容

- 课程起源
- 课程规划
- 五个有趣的现实问题
- 课程主体内容
- 注意事项 

# 课程主页

<http://course.pku.edu.cn>



**北大教学网**  
TEACHING AND LEARNING@PKU

热线 62767551  
邮箱 [course@pku.edu.cn](mailto:course@pku.edu.cn) English

[首页](#)
[北大课程](#)
[教改项目](#)
[实践与应用](#)
[培训与服务](#)
[关于教学网](#)
[统计信息](#)



**开学啦! 备战新学期**  
欢迎使用北大教学网

**请在此登录**

用户名:

密 码:

 访客登录 [登 录](#)

[正在直播](#) [马上点播](#)

**日程**

>> read more

**课程通知, 课后作业等**

more

02-14

□ 关于组织教师参加“教学新思路”教改项目第七期的通知 2011-12-12

□ 教师教育技术一级培训之一-Excel实战应用 2011-12-08

**新闻**

>> read more

□ 2011年北京大学教师教育技术一级培训圆满结束 2011-12-23

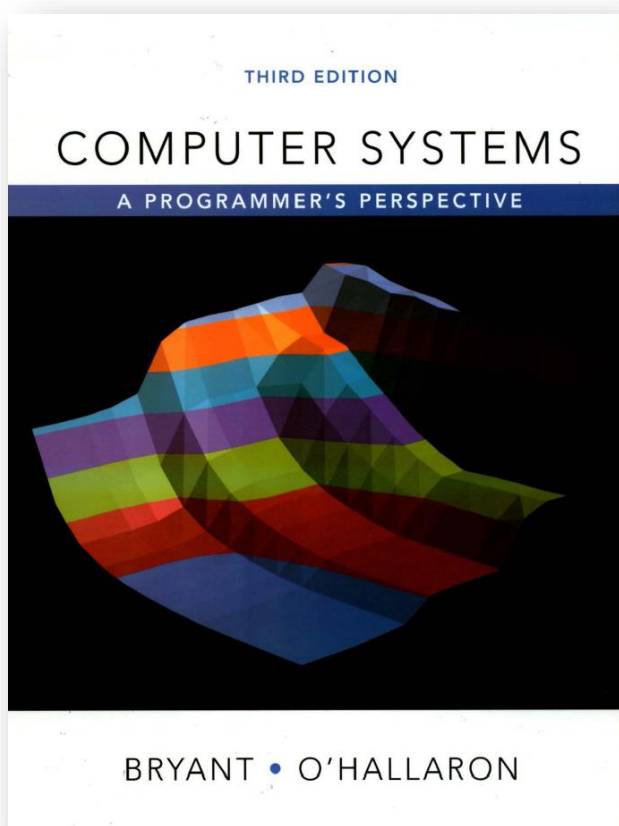
**教师视频**

 让文学研究伴学生成长

常规培训: 每周五14:00 电教403

# 课程教材

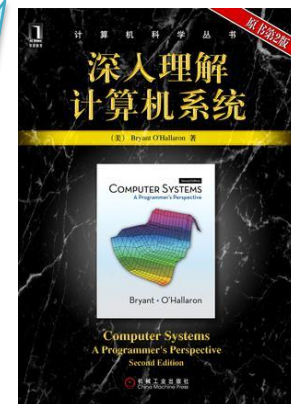
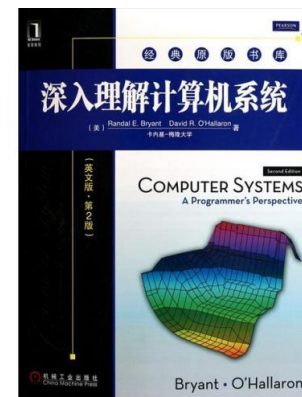
- **Computer Systems: A Programmer's Perspective (3<sup>rd</sup> Edition)**  
**深入理解计算机系统 (第3版)**
- **英文版作者: (美) Randal E. Bryant / David O'Hallaron**



教材第3版



这些是第2版



# 成绩评定

## 平时表现

小班课研讨表现、书面作业等

## 期中考试

笔试

30%

15%

15%

40%

## 实验题

系统评分+教师评定

## 期末考试

笔试

注：教学团队可能会根据实际教学情况对成绩评定比例进行适度调整

# 需要注意的问题

**Q: 为什么教学网的小班和安排的不一致?**

**A:** 根据往年情况, 开学前几周, 教学网选课名单可能会不稳定, 使用会有不便, 补选结束就可正常使用。

**Q: 小班是怎么安排的?**

**A:** 小班不用选课, 由教师统一安排分配, 预计周三会公布分班名单, 大班上和教学网。根据班号到指定教室上课, 不可自行换班。

**Q: 如果小班分班名单里没有怎么办?**

**A:** 少数同学可能会因为补选等原因, 没有出现在小班分班名单里。首先请随时关注教学网的更新通知, 如果课前还没有分到班, 可以在某个小班先参与, 待选课最终确定后, 小班名单也会最终确定。



