ICS 第 0x01 次小班课

2022 Fall, Class 12, TA: Yuxing Xiang

2022-09-07

自我介绍

EchoST
四川 成都

- 2020级计算机系本科生 向昱行
- 联系方式
  - 微信号：EchoStone20190405
  - E-mail：echost@stu.pku.edu.cn
  - QQ：2304969671（不常用）
- 爱好
  - 听歌，喜欢 J-Rock，沉迷ヨルシカ
  - 康美乐长期划水人
  - 最近对 DnD 挺有兴趣
  - 欢迎私下来唠嗑～

Introduction to ICS

What are we learning?

- 五个 Great Realities?
- **CSAPP** - Computer System: A **Programmer's** Perspective?

- 本课的内容定位：
  - 如果你是计算机领域刚起步的学习者，ICS将会为你初次揭开计算机系统的神秘面纱
  - 学会系统，才能更好地利用系统；ICS或许不能帮你成为一个系统设计者，但一定会把你变成更好的程序员

- 个人关键词：**Abstraction**

Introduction to ICS

An **enhanced** tour

```c
#include <stdio.h>
#include <assert.h>

int main() {
    // By Eric Cole
    // (IF YOUR CPU BRANCHES SLOWLY):
    unsigned int v = 65536;  // 32-bit value to find the log2 of
    register unsigned int r; // result of log2(v) will go here
    register unsigned int shift;
    r =     (v > 0xFFFF) << 4; v >>= r;
    shift = (v > 0xFF  ) << 3; v >>= shift; r |= shift;
    shift = (v > 0xF   ) << 2; v >>= shift; r |= shift;
    shift = (v > 0x3   ) << 1; v >>= shift; r |= shift;
                               r |= (v >> 1);
    assert (r == 16);

    printf ("r is %d\n", r);

    // Did you think this would break things?
    *((int *)&v - r * 10000) = 0xDEADBEEF;

    return 0;
}
```

Chap2 信息的表示和处理

整数/实数怎样表示?

代码与数据的底层储存?

一些Bit Hacks

- 编码规范本身是一种Abstraction

Introduction to ICS

An **enhanced** tour

```
 to find the log2 of
g2(v) will go here


r |= shift;
r |= shift;
r |= shift;
r |= (v >> 1);


?
```

```
00000000000006fa <main>:
 6fa:   55                      push   %rbp
 6fb:   48 89 e5                mov    %rsp,%rbp
 6fe:   41 54                   push   %r12
 700:   53                      push   %rbx
 701:   48 83 ec 10             sub    $0x10,%rsp
 705:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
 70c:   00 00
 70e:   48 89 45 e8             mov    %rax,-0x18(%rbp)
 712:   31 c0                   xor    %eax,%eax
 714:   c7 45 e4 00 00 01 00    movl   $0x10000,-0x1c(%rbp)
 71b:   8b 45 e4                mov    -0x1c(%rbp),%eax
 71e:   3d ff ff 00 00          cmp    $0xffff,%eax
 723:   76 07                   jbe    72c <main+0x32>
 725:   b8 10 00 00 00          mov    $0x10,%eax
 72a:   eb 05                   jmp    731 <main+0x37>
 72c:   b8 00 00 00 00          mov    $0x0,%eax
 731:   89 c3                   mov    %eax,%ebx
 733:   8b 45 e4                mov    -0x1c(%rbp),%eax
 736:   89 d9                   mov    %ebx,%ecx
 738:   d3 e8                   shr    %cl,%eax
 73a:   89 45 e4                mov    %eax,-0x1c(%rbp)
 73d:   8b 45 e4                mov    -0x1c(%rbp),%eax
 740:   3d ff 00 00 00          cmp    $0xff,%eax
 745:   76 07                   jbe    74e <main+0x54>
 747:   b8 08 00 00 00          mov    $0x8,%eax
 74c:   eb 05                   jmp    753 <main+0x59>
 74e:   b8 00 00 00 00          mov    $0x0,%eax
 753:   41 89 c4                mov    %eax,%r12d
 756:   8b 45 e4                mov    -0x1c(%rbp),%eax
 759:   44 89 e1                mov    %r12d,%ecx
 75c:   d3 e8                   shr    %cl,%eax
 75e:   89 45 e4                mov    %eax,-0x1c(%rbp)
```

Chap3 程序的机器级表示

C语言到可读机器语言的映射？不同控制结构的指令模式？过程的实现？数据的组织？程序安全性？

左边图上的一切细节到时候都会 make sense!

- 高级语言是机器语言的 Abstraction

Introduction to ICS

An **enhanced** tour

```
00000000000006fa <main>:
 6fa:   55                      push   %rbp
 6fb:   48 89 e5                mov    %rsp,%rbp
 6fe:   41 54                   push   %r12
 700:   53                      push   %rbx
 701:   48 83 ec 10             sub    $0x10,%rsp
 705:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
 70c:   00 00
 70e:   48 89 45 e8             mov    %rax,-0x18(%rbp)
 712:   31 c0                   xor    %eax,%eax
 714:   c7 45 e4 00 00 01 00    movl   $0x10000,-0x1c(%rbp)
 71b:   8b 45 e4                mov    -0x1c(%rbp),%eax
 71e:   3d ff ff 00 00          cmp    $0xffff,%eax
 723:   76 07                   jbe    72c <main+0x32>
 725:   b8 10 00 00 00          mov    $0x10,%eax
 72a:   eb 05                   jmp    731 <main+0x37>
 72c:   b8 00 00 00 00          mov    $0x0,%eax
 731:   89 c3                   mov    %eax,%ebx
 733:   8b 45 e4                mov    -0x1c(%rbp),%eax
 736:   89 d9                   mov    %ebx,%ecx
 738:   d3 e8                   shr    %cl,%eax
 73a:   89 45 e4                mov    %eax,-0x1c(%rbp)
 73d:   8b 45 e4                mov    -0x1c(%rbp),%eax
 740:   3d ff 00 00 00          cmp    $0xff,%eax
 745:   76 07                   jbe    74e <main+0x54>
 747:   b8 08 00 00 00          mov    $0x8,%eax
 74c:   eb 05                   jmp    753 <main+0x59>
 74e:   b8 00 00 00 00          mov    $0x0,%eax
 753:   41 89 c4                mov    %eax,%r12d
 756:   8b 45 e4                mov    -0x1c(%rbp),%eax
 759:   44 89 e1                mov    %r12d,%ecx
 75c:   d3 e8                   shr    %cl,%eax
 75e:   89 45 e4                mov    %eax,-0x1c(%rbp)
```

Chap4 处理器体系结构

电路和电器元件怎样实现指令?
什么是ISA?
换句话说，为什么机器语言被设计成这样？它们又是如何具体执行的？在这个层面是否有效率高低之分？

- 机器语言又可能是微指令的Abstraction

Introduction to ICS

An **enhanced** tour

```c
#include <stdio.h>
#include <assert.h>

int main() {
    // By Eric Cole
    // (IF YOUR CPU BRANCHES SLOWLY):
    unsigned int v = 65536;  // 32-bit value to find the log2 of
    register unsigned int r; // result of log2(v) will go here
    register unsigned int shift;
    r =     (v > 0xFFFF) << 4; v >>= r;
    shift = (v > 0xFF  ) << 3; v >>= shift; r |= shift;
    shift = (v > 0xF   ) << 2; v >>= shift; r |= shift;
    shift = (v > 0x3   ) << 1; v >>= shift; r |= shift;
                               r |= (v >> 1);
    assert (r == 16);

    printf ("r is %d\n", r);

    // Did you think this would break things?
    *((int *)&v - r * 10000) = 0xDEADBEEF;

    return 0;
}
```

Chap5 优化程序性能

利用现代编译器的特点优化程序效率

可以看成前面章节的综合运用；你会了解到左图代码中把循环拆开的意义（或者没有意义？），以及更通用的优化准则

- 本质是在利用Abstraction中非透明的特性

Introduction to ICS

An **enhanced** tour

```c
#include <stdio.h>
#include <assert.h>

int main() {
    // By Eric Cole
    // (IF YOUR CPU BRANCHES SLOWLY):
    unsigned int v = 65536;  // 32-bit value to find the log2 of
    register unsigned int r; // result of log2(v) will go here
    register unsigned int shift;
    r =     (v > 0xFFFF) << 4; v >>= r;
    shift = (v > 0xFF  ) << 3; v >>= shift; r |= shift;
    shift = (v > 0xF   ) << 2; v >>= shift; r |= shift;
    shift = (v > 0x3   ) << 1; v >>= shift; r |= shift;
                               r |= (v >> 1);
    assert (r == 16);

    printf ("r is %d\n", r);

    // Did you think this would break things?
    *((int *)&v - r * 10000) = 0xDEADBEEF;

    return 0;
}
```

Chap6 存储器层次结构

磁盘/内存/缓存/寄存器…
数据究竟被存储在哪?
你会学习到层次机构对计算机性能的影响，以及**缓存**的重要概念

- 深入系统为你提供的存储器模型的Abstraction

Introduction to ICS

An **enhanced** tour

```c
#include <stdio.h>
#include <assert.h>

int main() {
    // By Eric Cole
    // (IF YOUR CPU BRANCHES SLOWLY):
    unsigned int v = 65536;  // 32-bit value to find the log2 of
    register unsigned int r; // result of log2(v) will go here
    register unsigned int shift;
    r =     (v > 0xFFFF) << 4; v >>= r;
    shift = (v > 0xFF  ) << 3; v >>= shift; r |= shift;
    shift = (v > 0xF   ) << 2; v >>= shift; r |= shift;
    shift = (v > 0x3   ) << 1; v >>= shift; r |= shift;
                               r |= (v >> 1);
    assert (r == 16);

    printf ("r is %d\n", r);

    // Did you think this would break things?
    *((int *)&v - r * 10000) = 0xDEADBEEF;

    return 0;
}
```

Chap7 链接

多个源代码怎么被整合成一个可执行文件?
你有想过 `printf()`的实现是如何为你所用的吗?

从本章起转入更偏implementation的风格。

- 剖析“可执行文件”这一Abstraction

Introduction to ICS

An **enhanced** tour

```c
#include <stdio.h>
#include <assert.h>

int main() {
    // By Eric Cole
    // (IF YOUR CPU BRANCHES SLOWLY):
    unsigned int v = 65536;  // 32-bit value to find the log2 of
    register unsigned int r; // result of log2(v) will go here
    register unsigned int shift;
    r =     (v > 0xFFFF) << 4; v >>= r;
    shift = (v > 0xFF  ) << 3; v >>= shift; r |= shift;
    shift = (v > 0xF   ) << 2; v >>= shift; r |= shift;
    shift = (v > 0x3   ) << 1; v >>= shift; r |= shift;
                               r |= (v >> 1);
    assert (r == 16);

    printf ("r is %d\n", r);

    // Did you think this would break things?
    *((int *)&v - r * 10000) = 0xDEADBEEF;

    return 0;
}
```

Chap8 异常控制流

程序真的在不间断地执行吗?
为什么可以边刷树洞边听歌?
你知道 `printf()` 途中，你的程序会被内核“接管”吗?

你会首次接触到多个控制流；你会学习进程和信号等重要概念。

- 理解进程这一重要的 Abstraction

Introduction to ICS

An **enhanced** tour

```c
#include <stdio.h>
#include <assert.h>

int main() {
    // By Eric Cole
    // (IF YOUR CPU BRANCHES SLOWLY):
    unsigned int v = 65536;  // 32-bit value to find the log2 of
    register unsigned int r; // result of log2(v) will go here
    register unsigned int shift;
    r =     (v > 0xFFFF) << 4; v >>= r;
    shift = (v > 0xFF  ) << 3; v >>= shift; r |= shift;
    shift = (v > 0xF   ) << 2; v >>= shift; r |= shift;
    shift = (v > 0x3   ) << 1; v >>= shift; r |= shift;
                               r |= (v >> 1);
    assert (r == 16);

    printf ("r is %d\n", r);

    // Did you think this would break things?
    *((int *)&v - r * 10000) = 0xDEADBEEF;

    return 0;
}
```

Chap9 虚拟内存

为什么需要虚拟的内存空间?谁在使用?怎么映射到物理内存?

你会了解到左边代码中最后奇怪操作的合法性，以及更多烧脑（同时也巧妙）的设计和机制。

- 理解虚拟内存是如何对物理内存资源做Abstraction的

Introduction to ICS

An **enhanced** tour

```c
#include <stdio.h>
#include <assert.h>

int main() {
    // By Eric Cole
    // (IF YOUR CPU BRANCHES SLOWLY):
    unsigned int v = 65536;  // 32-bit value to find the log2 of
    register unsigned int r; // result of log2(v) will go here
    register unsigned int shift;
    r =     (v > 0xFFFF) << 4; v >>= r;
    shift = (v > 0xFF  ) << 3; v >>= shift; r |= shift;
    shift = (v > 0xF   ) << 2; v >>= shift; r |= shift;
    shift = (v > 0x3   ) << 1; v >>= shift; r |= shift;
                               r |= (v >> 1);
    assert (r == 16);

    printf ("r is %d\n", r);

    // Did you think this would break things?
    *((int *)&v - r * 10000) = 0xDEADBEEF;

    return 0;
}
```

Chap10 系统级I/O

程序是如何与文件交互的？
文件又是什么？
你知道 `printf()` 最终会调用 `write()` 吗？

你会学到从系统应用者角度出发，处理I/O的一般准则，以及文件描述符的机制。
从本章起的三章，深度止于应用而不是设计。

- 接触到文件这一重要 Abstraction

Introduction to ICS

An **enhanced** tour

Chap11 网络编程 &
Chap12 并发编程

从编程者的角度接触到网络和并发这两个重要概念。你分别会在计算机网络和操作系统课程中更加深入地探讨这两个话题。

- 初探IP/Socket/线程等等
  Abstraction…

Introduction to ICS

What are Labs? #1

- **Datalab (out?)**：玩转数据位级表示

  难度：★　　代码量：★

- **Bomblab**：拆除二进制炸弹，考验你对汇编代码的理解

  难度：★ ★　　代码量：★

- **Attacklab**：多种方式攻击可执行程序，体现程序安全性

  难度：★ ★　　代码量：★

- **Archlab**：处理器实验，同时优化ISA和程序

  难度：★ ★ ★　　代码量：★ ★

Introduction to ICS

What are Labs? #2

- **Cachelab**：内存访问性能优化实验，提升缓存命中次数

  难度：★ ★ ★ ★　　代码量：★ ★ ★

- **Tshlab**：简易Shell实验，实现你自己的Shell

  难度：★ ★ ★ ★ ★　　代码量：★ ★ ★

- **Malloclab**：动态内存管理实验，实现自己的 malloc()

  难度：★ ★ ★ ★ ★　　代码量：★ ★ ★ ★ ★

- **Proxylab**：网络代理实验，实现自己的代理

  难度：★ ★ ★ ★　　代码量：★ ★ ★ ★ ★

Introduction to ICS

More about Labs #1

- **时间安排：**
  - Handout
  - Due
  - Deadline
  - Graceday

周次	周一周三大班	节次	主题		LAB日程
一	9月5日	1	Overview		
	9月7日	2	Bits and Bytes/Integers		L1 (datalab) out
二	9月12日		中秋节放假		
	9月14日	3	Floating Point		
三	9月19日	4	Machine Prog: Basics		
	9月21日	5	Machine Prog: Control		L2 (bomblab) out
四	9月26日	6	Machine Prog: Procedures		
	9月28日	7	Machine Prog: Data		
五	10月3日	8	Machine Prog: Advanced		L3(attacklab) out
	10月5日	9	Processor Arch: ISA&Logic		
六	10月10日	10	Processor Arch: Sequential		L4 (archlab) out
	10月12日	11	Processor Arch: Pipelined		
七	10月17日	12	The Memory Hierarchy		
	10月19日	13	Cache Memories		L5 (cachelab) out
八	10月24日	14	Program optimization		
	10月26日	15	期中考试		

Introduction to ICS

More about Labs #2

- **关于抄袭**
  - 课本代码随便用
  - 请不要查看其余任何代码，包括同学/网上/往届代码
  - 同学之间可以在设计层面进行讨论（主要针对后半期）
- 实在做不出来？
  - 可以找我要提示
  - 千万不要面向 CSDN/Github/StackOverflow/... 编程!
  - 会有**查重**
- Start early! 意想不到的事情很多!
- Plan ahead! 用好每个 lab 的 Writeup，最好写之前能有规划!
- 多数 lab 提交次数有限制，取最后一次提交为成绩
- 每个 lab 占总评 3%，合理取舍！
  - 未必都要肝到满分，偶尔一个 90/100 对总评没有影响
  - 比如 Malloclab 满分极难

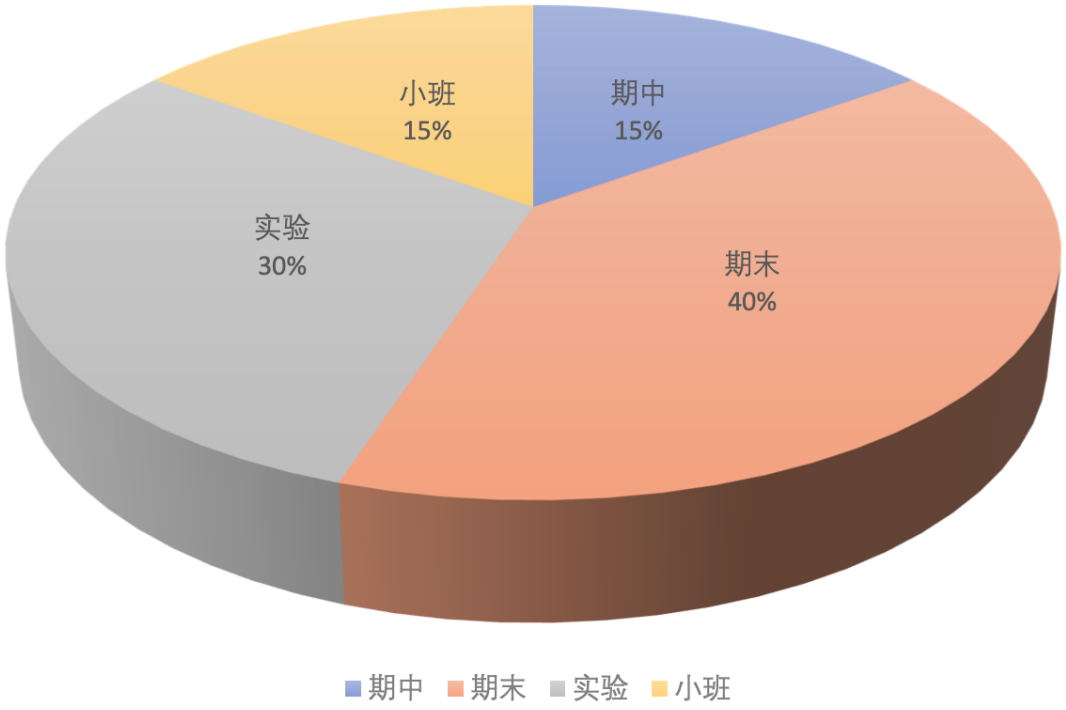
Introduction to ICS

More about Labs #3

- **最终测试统一在 autolab.pku.edu.cn (Ubuntu 18.04 LTS)**
  - 测试脚本本身都会发放，可以本地自测
  - 每人都会有 class machine 账号，利用VScode + remoteSSH 可以远程开发
  - 部分跟 Performance 相关的测试可能因为服务器压力有波动，以 autolab 结果为准
  - 遇到提交问题，我可以帮你查看后台日志
  - 请不要（无意识）攻击服务器，维护的助教和你都不会开心的
  - Lab 刚发布时可能有各种小问题，可以等半天再开做
- **Life is more than Lab! 有的时候暂时的放下意味着柳暗花明**
- **Lab 的本意不是为难大家；用好课本和 Writeup，多数问题不在话下**
- **我计划在每个 Lab 放出后做一个 Lab Preview，帮助大家更快上手**

Introduction to ICS

Grading



Introduction to ICS

Seminar 小班安排

- **回课/专题讨论**
  - 小班课评分的**主要**内容。具体占比待定。
  - 分派到人；我稍后会发出腾讯文档，里面会提到每次回课大概的章节内容和小班节次，可以根据自己意愿占坑。先到先得，剩下的由我分配。
  - **形式**可以任意（slides/板书/现场上机演示…）；**内容**则需要围绕相应课程内容，包括书本和大课。
    - 以 topic 形式组织；需要覆盖相应章节最重要/疑难的内容，以及一两个稍微拓展的探索话题
    - 本意是杜绝 PPT reader！
    - 具体讨论什么 topic，我会提出建议；更欢迎大家主动向我提出想讲的 topic（完全可以是你自己想学清楚的内容，我可以陪你一起学会，再由你教给大家）！
  - 需要主讲人在每周日前形成草稿（可以是 high-level 的），周日、周一约我讨论草稿。请主动联系，不要等到我来 check on you!
  - 主讲期间欢迎抽同学回答问题！
  - * 不建议在讨论中加入过多做题环节

Introduction to ICS

Seminar 小班安排

- **出勤**
  - 小班课评分的次要内容。具体占比待定。
  - 无特殊理由迟到十五分钟以上一次扣0.5分；无特殊理由缺勤扣1分。扣完本部分分为止。
  - 除了因为不在校/个别特殊理由外，不提供线上方式。

- **助教补充**
  - 我来补充讲解的环节；本意是从一个经验性的/应试的角度出发，补充回课同学容易忽略的地方。
  - 包含适量的**习题练习**（也因此不需要回课同学准备太多题目）；是否计分？
  - 也相当于答疑环节，有任何问题可以当场提问！

- **Lab Preview & Lab Summary**
  - 由我来对 Lab 做简单的导引和总结，点评一些关键点和出彩解法
  - 欢迎自愿报名 Lab Summary，**不计分**，但留给你炫耀/抱怨的空间x

Lab Preview

Datalab

- TBA

- 环境配置：已经在tutorial里cover
- 介绍 Writeup 的用法
- 简介 Lab 内容

Shell Overview

MIT Missing-Semester

- **Shell是什么?**

一个文字界面，可以执行程序、代码，甚至控制或监测计算机的运行状态。图形界面、声音控制、手势控制等等易用而且更现代的接口使得计算机能够被大众所用。但大多数**底层的功能**和**高效的执行**还是需要基于shell。
Shell顾名思义是一层壳，外面是用户，里面是内核或者宽泛地说是操作系统的一部分。

- **演示时间**
  - 如果你有 Linux 环境，可以试着跟我一起操作
  - 演示流程基于 MIT Missing-Semester 第一节课内容；如果你没有跟上，可以自己再去看一遍！

Next Class…

Bits/Bytes/Integers

- **建议**
  - 本章是ICS唯一数学性比较强的章节。可以从数学角度对书中论证来仔细验证一下，有助于加深理解。
  - 开学这段时间是ICS最轻松的时期。如果你觉得目前的内容已经掌握得很熟练，不妨稍微往后自学一些（程序的机器级表示）。
  - 稍后我会在群里分享一个北大网盘链接。小班课的资料都会放在网盘里，包括我的所有slides，往年大班slides，练习题，以及往年题（答案仅供参考，事实上就是有错，欢迎找我讨论；不过建议把近两三年的题目留出来考前做～）

Some Final Tips

How to learn ICS

- **重视书本。**课堂内容高度概括，硬核技术和细节都需要自己课下理解掌握！你可能需要提前阅读课堂对应的章节。
- **尽力做练习题。**书上的练习题不见得都那么有意思，但是做练习题可以促进你对书本的理解。一味硬啃往往是看不进去的。
- **欢迎提问。**助教的职责就是帮助大家解决学习过程中的任何疑问。另一方面，也希望你能掌握提问的艺术：提供足够的信息和描述，并铭记两个准则：RTFM和STFW！
- **适应风格变化。**这门课在半期前和半期后的内容风格变化不小，和教材编排也是一致的。希望你做好各方面的准备。
- **应试的现实。**相比CMU原装，北大ICS的课程因为优秀率限制而素有恶名。这方面，去年助教优秀的命题给我们助教开了个好头；但你也要计划好在备考上投入不小的努力，并重视做题环节。
- **一些个人见解**

Any Questions?

Some Final Tips

… also for myself

最后，我也是第一次当助教，并且现在还因为疫情原因暂时不能跟大家线下见面。各方面可能都有做得不够周到的地方，还请大家不吝提出意见/建议！

有一个根本准则是我希望能坚持的：**重视基础**。我可能会因为过来人的角度而忽视大家初学时的一些困惑，也可能因为自己稍微多学的一点知识而不够贴近这门课程的实际。一旦你觉得我的进度安排不够合理，或者是你觉得我一直在讲花的，请一定打断我的表演。

希望能陪大家度过一个愉快、充实的学期 :）

Thanks for Listening