

操作系统A

Principles of Operating System

北京大学计算机科学技术系 陈向群

Department of computer science
and Technology, Peking University

2020 Autumn

大纲

- ▶ 本课程相关的基本信息
 - ▶ 学习资源、课程形式、学习要求、评价体系
- ▶ 教学思路
- ▶ 如何学好这门课?
- ▶ 认识操作系统
 - ▶ 操作系统的地位、名称的演化
- ▶ 操作系统做了什么?
- ▶ 操作系统的主要工作

课程说明 课程目标 评分体系.....

课程基本信息

课程形式

- ▶ 主课
- ▶ 操作系统源代码分析 (XV6)
- ▶ 习题课/课堂讨论课

选做项目：

- ▶ 话剧表演。。。。
- ▶ 课堂展示。。。。
- ▶ 课外实践。。。。

MOOC, 操作系统原理, 华文慕课平台

个性化设计

关于话剧

▶ 目的

通过 **话剧的形式** 对 操作系统课程
的 **内容** 有 **更深入的理解**

▶ 做法

▶ 小组形式 通过讨论方式 共同完成剧本

▶ 围绕 若干知识点（概述、运行环境、
进程线程模型、内存管理、文件系统、
设备管理、死锁等等）

▶ 表演

▶ 事件节点

▶ 组队报名，剧本创作，排练，登场演出

A Small Play

- ✓ 剧本写作
- ✓ 导演/剧务
- ✓ 演员

源代码分析：xv6
















- ▶ 一个真正能在硬件上运行的简单操作系统
- ▶ 基于x86架构的计算机系统上
- ▶ 支持对称多处理器（SMP）的类UNIX的教学用操作系统
- ▶ 起源于MIT
- ▶ <http://pdos.csail.mit.edu/6.828/2012/xv6.html>

技术要求

- ▶ Unix Version 6
- ▶ Architecture: the PDP-11 and the Intel x86
- ▶ AT&T汇编和GCC内联汇编
- ▶ 编译链接等相关技术
- ▶ 其他可选了解项目
 - ▶ Git
 - ▶ Gccb编译调试工具链
(gcc/ld/make/gdb/objdump/objcopy)

功能模块

- 目录下有多个文件，由C语言和汇编语言写成（为什么？）

 asm.h	2013/9/8 22:11	H 文件
 bio.c	2013/9/8 22:11	C 文件
 bootasm.S	2013/9/8 22:11	Assembler Source
 bootmain.c	2013/9/8 22:11	C 文件
 buf.h	2013/9/8 22:11	H 文件
 BUGS	2013/9/8 22:11	文件
 cat.c	2013/9/8 22:11	C 文件
 console.c	2013/9/8 22:11	C 文件
 cuth	2013/9/8 22:11	文件
 defs.h	2013/9/8 22:11	H 文件
 dot-bochsrc	2013/9/8 22:11	文件
 echo.c	2013/9/8 22:11	C 文件
 elf.h	2013/9/8 22:11	H 文件
 entry.S	2013/9/8 22:11	Assembler Source
 entryother.S	2013/9/8 22:11	Assembler Source

功能模块

- ▶ 系统调用
- ▶ 进程调度
- ▶ 内存管理
- ▶ 中断处理
- ▶ 文件系统
- ▶ I/O管理（设备管理）

源代码阅读要求

- ▶ 根据上课内容及进度，阅读XV6对应代码，了解XV6的工作原理和实现机制
- ▶ 通过阅读相关代码，并自行查阅相关资料，在理解XV6工作原理的基础上撰写代码阅读报告
- ▶ 报告内容：原理课讲授的机制与实现和XV6机制与实现的比较，从性能、实现复杂性、硬件支持和可扩展性等方面进行讨论
- ▶ 选择部分同学在课堂上展示源代码阅读成果

课程Lab——Nachos实习

- 完成 6 个Lab的实习，选项完成2个Lab
- 课上 以讨论为主
 - 重要知识点讨论
 - 实习过程疑难问题讨论
- 课下
 - 完成实习任务
 - 撰写实习报告

Nachos简介

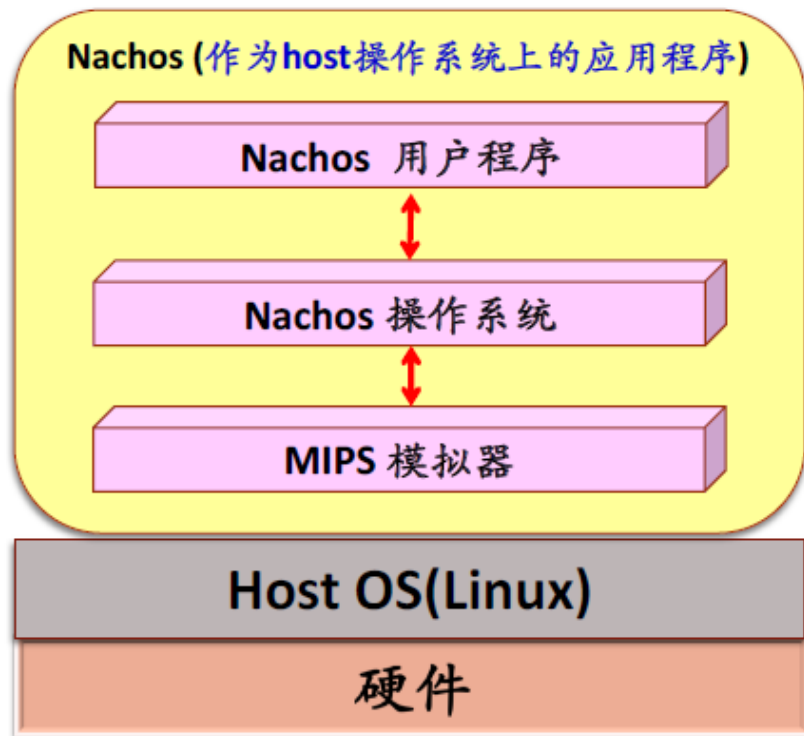
- **Nacho (Not Another Completely Heuristic Operating System)**
 - 一个用于教学的操作系统，由UC Berkeley开发
- 提供了操作系统的框架
 - 线程、用户进程、虚拟存储、中断驱动的I/O
- **Nachos还包含了硬件的模拟**
 - MIPS处理器、控制台、时钟、中断
- 实习任务——对其进行完善和修改
 - 线程、调度、同步机制、文件系统、内存管理、系统调用
 - Shell实现、通信机制（选做）

Nachos历史

- 1992年由UC Berkeley开发
 - Thomas E. Anderson, Wayne A. Christopher, Steven J. Procter
 - Written in C++ for MIPS
 - Berkeley、New York University等多所大学使用
- 2000年，Daniel Hettena(Berkeley)开发了Nachos 5.0 Java版
- 2004年，Stanford将Nachos移植到x86系统上，开发了pintos

Nachos体系结构

- Nachos本身是Linux宿主主机
的一个进程
 - Nachos包含一个MIPS虚拟机，
用于执行用户程序
- 用户程序是c程序经过交叉编译
后生成的MIPS可运行的程序
 - Nachos Kernel负责用户程序
的装载以及在MIPS模拟器
上的执行



Nachos运行原理

- Nachos的应用程序是MIPS指令格式的
 - 在x86平台上使用交叉编译生成
- Nachos Kernel和硬件模拟器运行在host OS上
 - Nachos Kernel实现系统服务
 - MIPS模拟器模拟了硬件行为，并且执行用户程序的每条指令
 - 用户程序可以触发系统异常，模拟器接收到异常后，转到kernel处理异常

用 户 程 序				
	线程管理	网络协议	文件系统	虚拟内存
终端设备	时钟	网络	磁盘	
中 断 系 统				指令解释和内存模拟

Nachos系统源代码树结构

- code 目录
 - threads 内核线程
 - fileys 文件系统
 - userprog 用户进程
 - machine 机器硬件模拟
 - vm 虚拟内存
 - network 网络系统
 - Makefile 工程makefile
 - Makefile.common

为什么要完成Lab?

- 纸上得来终觉浅，绝知此事需躬行

Learn OS concepts by coding them!

- 操作系统
 - 体现了软件中**核心的工程性技术**
 - 尽管理论相对成熟，但从工程实施和维护来说，即使对很多科班出身的人来说也是个黑盒子
- 操作系统实习
 - 通过自己动手写一个操作系统，或是在**已有框架基础上完善一个操作系统**
 - 能对操作系统理论有更深刻和感性的理解，很好提升专业素养

课程Lab要求——提升以下能力

□提炼总结能力

- ✓设计与实现解决方案

□动手能力

- ✓工具的使用、源代码阅读、调试技术、版本管理

□表达能力

- ✓阐述观点、回答问题
- ✓沟通交流、团队合作

□个人自我管理能力的

课程Lab要求

- 在原有框架基础上完成一个教学操作系统Nachos
 - 实践操作系统原理课学到的知识
 - 掌握操作系统的全局设计
 - 掌握操作系统各个重要模块（关键知识点）的设计与实现
- 掌握Nachos开发过程中需要使用的环境和工具
 - Linux环境、交叉编译工具、调试工具(gdb)
 - Linux下代码辅助阅读工具(ctags、cscope、source insight)
 - 版本控制工具(Git、CVS)

通过Lab, 你会得到什么?

- 对操作系统工作原理更深刻、更专业的理解
- 成就感、精神愉悦
- 获取分析、解决问题的信心和方法
- 对工程、编程语言的新的理解
- 自学能力的检验、**自我管理能力的考查**
- 一段可以拿出手的项目经历

Lab报告要求

- 按时提交，独立完成；逻辑清晰，文字通顺；书写规范，排版美观
- 提交报告请采用pdf或doc、docx格式
- 文件名请采用格式：学号_姓名_Lab名实习报告
- 报告内容及文档格式请参考报告模板，模板可以在教学网上下载
- 可以使用图片和文字来说明代码的正确性，不要大段大段的贴代码
- 不要凑字数，多花点时间总结和反思你所完成的内容，不仅可以让你加深对知识的理解，也会让你的报告获得更好的分数
- 对于报告的一点建议，报告不一定放到所有任务的最后来写，在做的过程中有什么问题或感想当时就可以记录下来，作为报告的素材，以免最后记不清了再去想象。
- 欢迎对课程提出任何意见或建议

相关资源

- (1) Nachos源码、Nachos介绍
源自<http://www.cs.washington.edu/homes/tom/nachos/>
- (2) [A Road Map Through Nachos.pdf](#), 给出了Nachos源码的概要分析
- (3) vagrant方法搭建资源地址
<https://disk.pku.edu.cn:443/link/B61F55C00976BDE7DE22054EE85ACD86>
- (4) 源码搭建资源地址
<https://disk.pku.edu.cn:443/link/F58D0E51955319CCF8140705025921BE>, nachos

课程目标

课程目标

Operating Systems are to Computer Science what mathematics is to engineering

- ▶ 掌握操作系统的基本概念、功能组成、系统结构及运行环境
- ▶ 熟悉并运用操作系统工作原理、设计方法和实现技术
- ▶ 了解操作系统的演化过程、发展研究动向、新技术以及新思想
- ▶ 理解各种有代表性的、典型的操作系统实例
- ▶ 培养发现问题、界定问题、解决问题、评估结果的基本能力，成为创新型人才
- ▶ 为后续课程打下良好基础，为后续发展奠定基石

课程平台

- ▶ 北大教学网

course.pku.edu.cn

- ▶ 华文慕课“操作系统原理课程”

<http://www.chinesemooc.org/mooc/4747>

授课教师及助教

- ▶ 主讲教师：陈向群
cherry@sei.pku.edu.cn
地点：理科1号楼1429

office hours
周四
4:00-5:00

- ▶ 助教：马辛宇、郭思敏、贺宁宇、张少坤
课程邮箱：pkuos2020@163.com
地点：理科1号楼1435

参考书目

► 参考书目

► Modern Operating System (4th Edition) (Andrew S. Tanenbaum) 现代操作系统

► Operating Systems Internals and Design Principles (William Stallings) 7th Edition
操作系统—精髓与设计原理

► Operating System Concept (Abraham Silberschatz) 7th Edition 操作系统概念

► Operating Systems: Three Easy Pieces

Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau

<http://pages.cs.wisc.edu/~remzi/OSTEP/>

其他学习资源

- ▶ **Windows内核原理与实现** 潘爱民 电子工业出版社
- ▶ **Linux内核设计与实现** [美] R.Love著 陈莉君译 机械工业出版社
- ▶ **Orange S:一个操作系统的实现** 于渊 著 电子工业出版社
- ▶ **观止：微软创建NT和未来的夺命狂奔** [美] G.Pascal Zachary, 张银奎等译 机械工业出版社
- ▶ **程序员的自我修养—链接、装载与库** 俞甲子 石凡 潘爱民 著 电子工业出版社
- ▶ ○ ○ ○ ○ ○ ○

其他学习资源

- ▶ **独辟蹊径品内核：Linux内核源代码导读** 李云华编著 电子工业出版社
- ▶ **Linux内核设计的艺术** 新设计团队 机械工业出版社
- ▶ **深入Linux内核架构** [德] Wolfgang Maurer 郭旭译 人民邮电出版社
- ▶ **Linux内核修炼之道** 任桥伟 编著 人民邮电出版社
- ▶ ○ ○ ○ ○ ○ ○ ○

考试信息

- ▶ 期中考试

2020年11月11日（暂定）

- ▶ 期末考试

2021 年01月18日 下午

课程成绩评定

- ▶ Labs及源代码阅读 30%
- ▶ 期中考试 20%
- ▶ 期末考试 50%

- ▶ 加分策略 (2 - 10分)
 - ▶ 话剧
 - ▶ 课程展示
 - ▶ 课外实践

如何学好“操作系统”这门课程？

与同学们交流、分享

为什么要学操作系统?

知乎

► 开眼界

作为最常见的复杂软件，操作系统包含了**时序与分时**等时间相关的经典案例，又是**接口与抽象**方面的极佳例子，涵盖了常见软件开发中所可能遇到的大部分场景。

► 打基础

真正做起工程就会知道，很多很多问题是操作系统相关的。

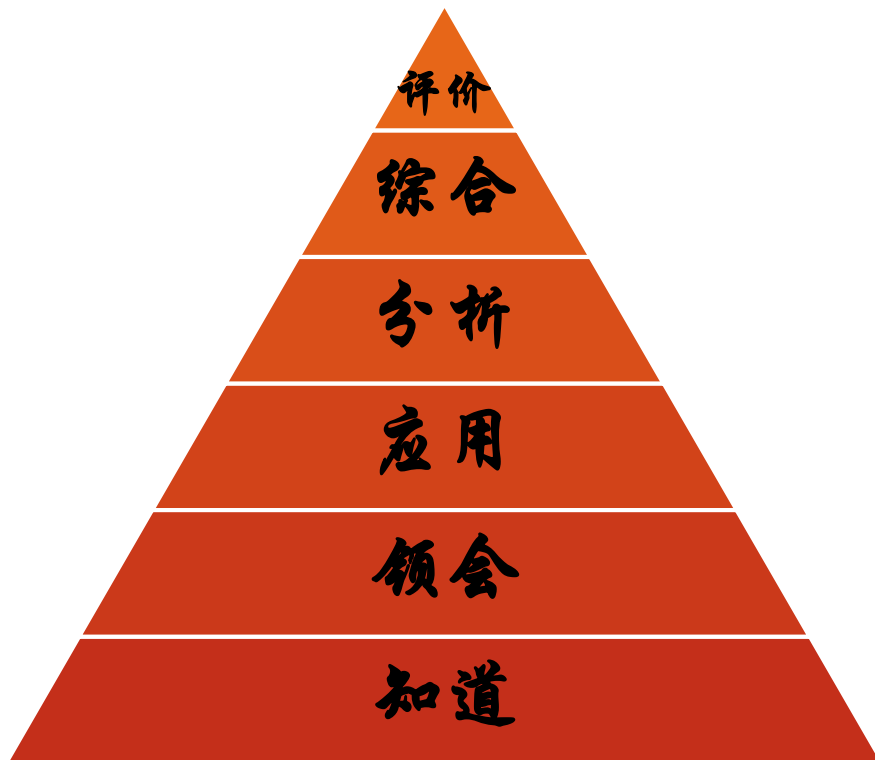
应用层开发的确只需要接触冰山在海面上的可见部分；但这只够你开发一些蹩脚的软件；冰山藏在海面下的9/10，和冰山的可见部分毕竟是一体的：浮于表面的软件同样会影响冰山的不可见部分、并被冰山的不可见部分影响。如果没有基了解，当冰山的不可见部分透过可见部分坑到你时，你绝没能力为这些蹩脚软件debug。

大学的理念

如果没有分析、归类和关联的过程，即使学习再多的知识，心智也谈不上扩展，头脑也算不得开窍，或者说，也称不上具有综合的理解力。

知识要扩展，就不能是仅仅被动地即不假思索地接收一堆新的思想观念，而是对涌现的新的思想观念，要能够及时并积极地思考，要批判的审视这些思想，主动地接近这些思想，从主体的角度去体验这些思想。这是一种构造性的思维活动，它将我们获得的知识转化为有条理、有意义的思想；它使我们的知识客体成为我们自己的主观知识，通俗地说，就是将我们接收的事物加以消化，使之与我们已有的思想融为一体；没有这个过程，知识就不会随之而扩展。各种观念进到头脑，如果不把一种观念与另一种观念比较，并使之系统化，就没有知识扩展可言。我们不仅学习，而且将所学的与已知的进行对照，只有这样，我们才会感到心智在生长、在扩展。

一般性学习目标



布鲁姆 教育目标层次 模型

层次	举例（高速缓存）
知道	高速缓存的概念
领会	描述某种缓存的设计思想
应用	高速缓存的具体实现、场景使用后的结果
分析	在某场景下对某高速缓存进行性能分析
综合	在特定场景下，综合各要素，设计高速缓存
评价	评价高速缓存在整个系统中的作用

ICS课与本课程的关系

- ▶ 存储器层次结构

- ▶ 局部性原理

Cache, 在内存缓存磁盘块 (虚存), Web Cache

- ▶ 异常控制流

中断、陷阱、故障、系统调用、内核/用户模式

- ▶ 进程、线程

- ▶ 信号

- ▶ 虚存

页表、TLB、地址翻译 (地址转换)

内存映射机制

- ▶ 并发、死锁

典型名词术语 (e.g.)

Multiprogramming	PCB	Starvation	Page replacement
Time-sharing	Context switch	Preemptive	Temporal locality
Booting	Thread	Priority	Programmed I/O
Protection	Multithreading	Deadlock	DMA
Privileged instruction	Scheduling	Wait-for graph	Buffer
Exception	Synchronization	Virtual memory	RAID
Fault	Data race	Paging	Disk cache
System call	Mutual exclusion	Virtual address	File
Interrupt	Critical section	Swapping	Directory
Process	Atomic operation	Page table	Device driver
Execution state	Lock	TLB	Page fault
Address space	Semaphore	Shared address space	Spatial locality

教学思路 (1/2)

► 四个层面

- ✓ 操作系统安装、使用（本课不涉及）
- ✓ 操作系统原理
 - 概念、技术、工作机制
- ✓ 操作系统实例
 - (Windows、Linux、Unix、Android)
- ✓ 实现一个操作系统 (Lab)

教学思路 (2/2)

不是 如何使用操作系统

而是 理解操作系统如何工作

- ▶ 操作系统的内部工作方式
- ▶ 操作系统实现涉及的 **数据结构和算法**
- ▶ 设计、开发操作系统过程中的 **问题、解决方案和折中权衡**
- ▶ 操作系统中的 **典型技术及应用**

如何学好这门课?

◎ 几点建议

- 读透教材
- 积极、主动阅读参考书
- 认真、按时完成作业、实践练习
- 思考, 讨论, 提问
- 及时沟通、反馈

➤ 快乐 de

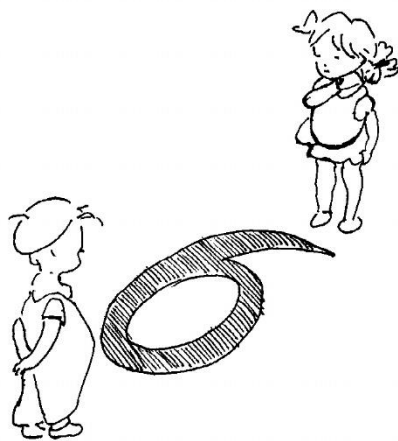
➤ 差异化

➤ 系统观

➤ 输入 → 处理 → 输出

➤ 观察 → 思考 → 提炼 → 沉淀

90%积累 / 10%创新

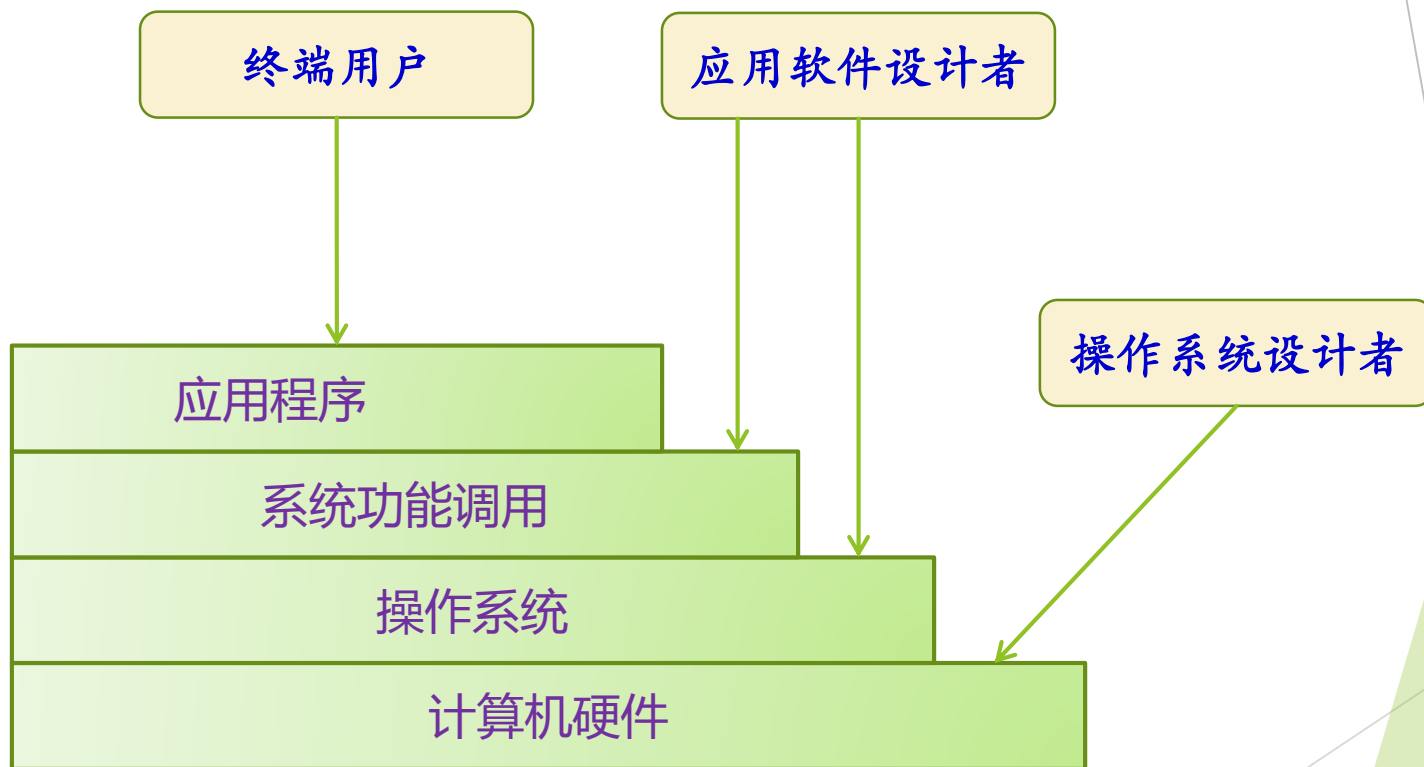


横看成岭侧成峰

OS的地位 名称的演化.....

认识操作系统

操作系统的地位



名称的演化

从
Supervisor
到
Operating

- ▶ 监控（督）程序（系统）(Monitor)
- ▶ 执行系统（程序）(Executive System(program))
- ▶ 控制系统（程序）(Control System (program))
- ▶ 管理程序(Supervisor, Supervisory System)
- ▶ 核心程序(Kernel)
- ▶ 操作系统(Operating System)



功能支持 提供服务 状态切换

操作系统做了什么？

操作系统做了什么? (1/4)

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    puts("hello world");
    return 0;
}
```

操作系统做了什么？ (2/4)

➤ 用户告诉操作系统执行helloworld程序(如何告知？)

➤ 操作系统：找到helloworld程序的相关信息，检查其类型是否是可执行文件；并通过程序首部信息，确定代码和数据在可执行文件中的位置并计算出对应的磁盘块地址（文件格式？）

➤ 操作系统：创建一个新的进程，并将helloworld可执行文件映射到该进程结构，表示由该进程执行helloworld程序

➤ 操作系统：为helloworld程序设置CPU上下文环境，并跳到程序开始处（假设调度程序选中hello程序）



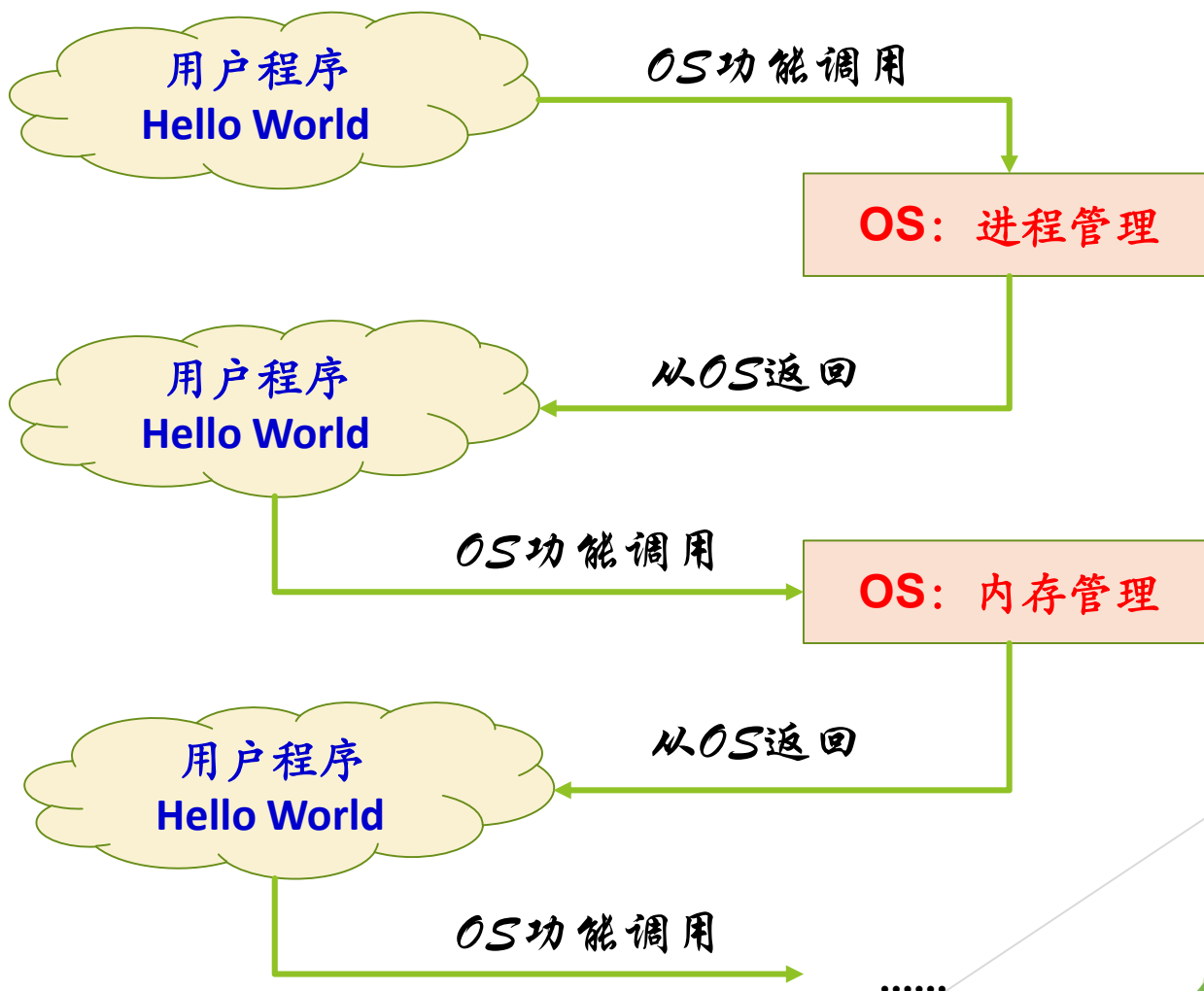
操作系统做了什么? (3/4)

- 执行helloworld程序的第一条指令，发生缺页异常
- 操作系统：分配一页物理内存，并将代码从磁盘读入内存，然后继续执行helloworld程序
- helloworld程序执行puts函数（系统调用），在显示器上写一字符串
- 操作系统：找到要将字符串送往的显示设备，通常设备是由一个进程控制的，所以，操作系统将要写的字符串送给该进程

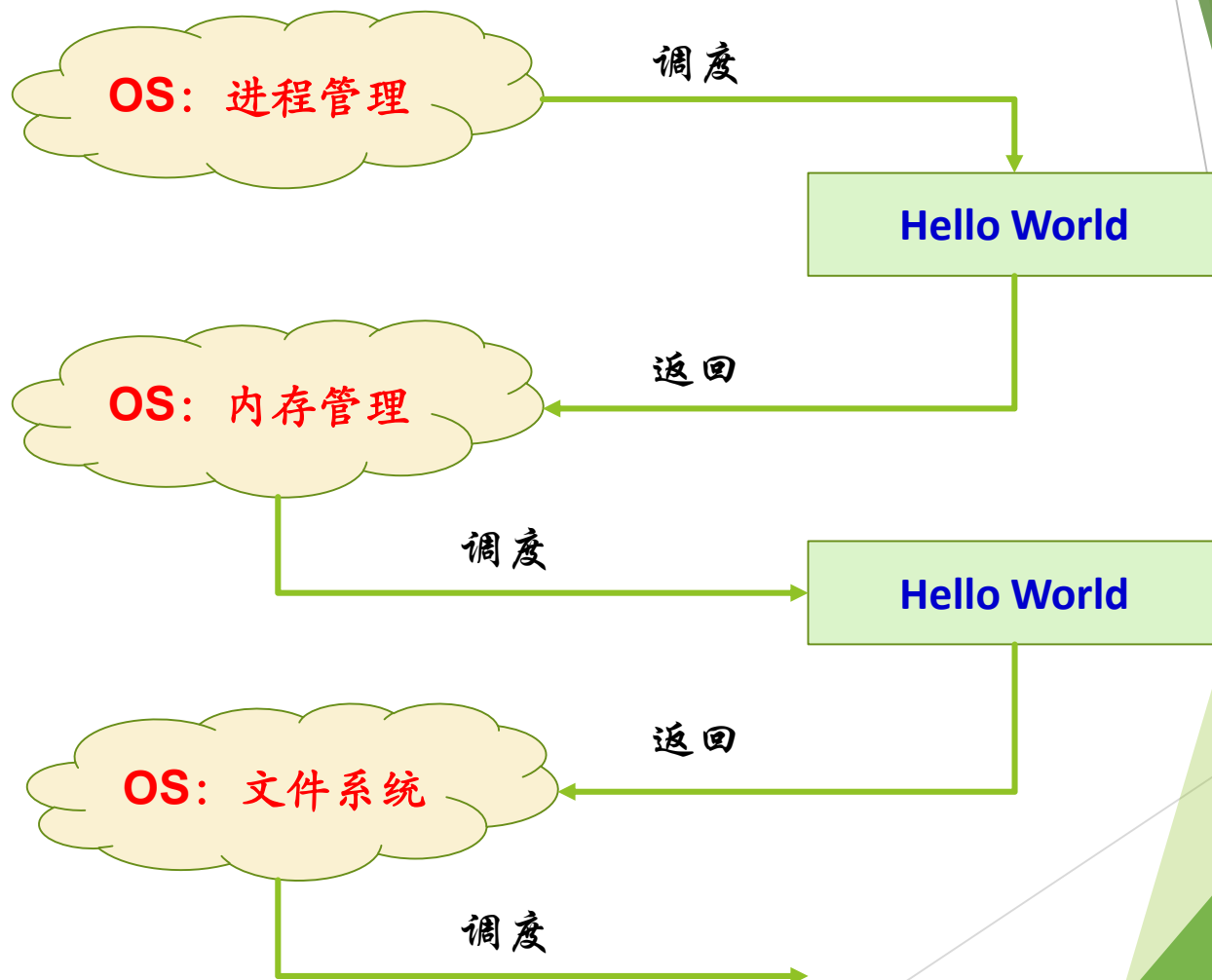
操作系统做了什么？(4/4)

- 操作系统：控制设备的进程告诉设备的窗口系统它要显示字符串，窗口系统确定这是一个合法的操作，然后将字符串转换成像素，将像素写入设备的存储映像区
- 视频硬件将像素转换成显示器可接收的一组控制/数据信号
- 显示器解释信号，激发液晶屏
- **OK!!! 我们在屏幕上看到了“hello world”**

从上述步骤中得到什么启示?



换个角度?



操作系统的主要工作

- ▶ 程序的执行
 - 启动程序、执行程序以及程序结束的工作
- ▶ 完成个性的工作（与硬件有关）
- ▶ 完成共性的工作
 - 易于使用，基本服务，统一性
- ▶ 性能、安全、健壮性等问题

个性——硬件相关 (1/3)

应用程序

----- 虚拟机器界面

操作系统

----- 物理机器界面

硬件



- ✓ 假如没有操作系统，怎样将目标代码送给硬件？
 - ✓ 怎样输出打印结果？
- 人们将对二进制程序操作 从二极发光管读答案

硬件相关 (2/3)

例如：实现代码中的存储器物理地址，或者对设备接口寄存器和设备接口缓冲区的读写等等

- ▶ 实现该工作的过程代码和硬件因素密切相关，即需要设置与测试、使用物理地址、设备接口寄存器等
- ▶ 硬件相关必然复杂繁琐、代码量大
- ▶ 硬件相关的工作，其实现代码不通用

硬件相关 (3/3)

- ▶ 有下列两个要求，你觉得哪一个更简单？
从文件读数据块 和 移动磁头、等待放下

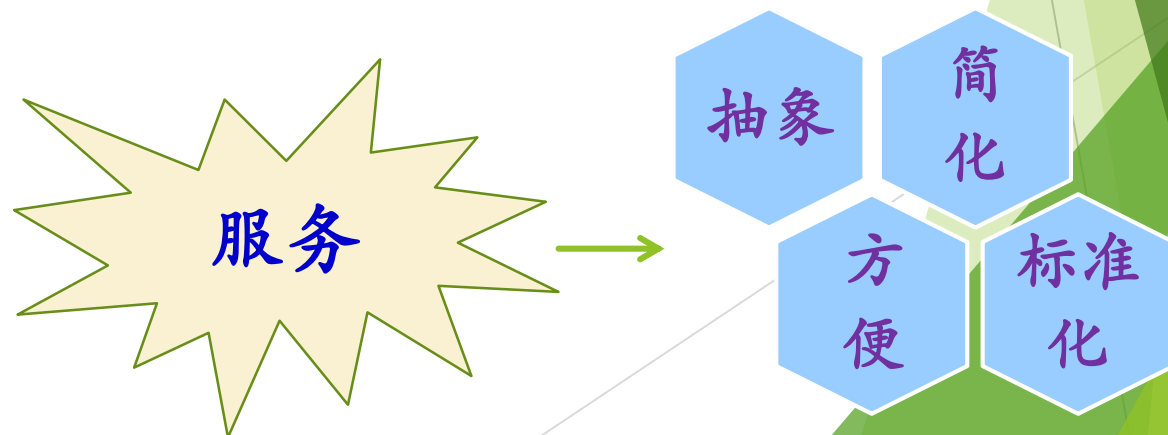
例子：软盘I/O操作

- 控制芯片**NEC PD765**：16条命令
- 每一条命令向一个设备寄存器装入长度从1到9字节的特定数据（读写数据、移动磁头臂、格式化磁道，及初始化、检测状态、复位、校准控制器及设备等）
- 以**READ**为例：13个参数
 - ✓ 要读取的磁盘块地址、磁道的扇区数、物理介质的记录格式、扇区间隙、对已删除数据地址标识的处理方法
 - ✓ 操作结束时，控制器芯片在7个字节中返回23个状态及出错字段
 - ✓ 软盘程序员还要保持注意步进电机的开关状态

共性

任何一个程序都需要的、最基本的工作

它们 具有共性、工作过程相同、与具体应用无
直接关系（即与用户所关心的应用目标无直
接关系）



应用软件与现实硬件之间的软件

- ▶ 硬件抽象，可移植性
- ▶ 有限变为无限（接近）
- ▶ 提供保护

一台等价的扩展机器（虚拟机），比底层硬件更容易编程



操作系统

硬件

作业0

► 请调研教材

Operating Systems: Three Easy Pieces

Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau

<http://pages.cs.wisc.edu/~remzi/OSTEP/>

简要概括总结一下该教材。

► 完成实现Lab的准备工作

- 1、了解Nachos运行原理
- 2、熟悉Nachos系统源代码树结构
- 3、搭建好完成Lab的实验环境

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic look.

Thanks

The End