通过ucore学习Linux操作系统内核原理与设计实现

陈渝 向勇
清华大学计算机系
yuchen@tsinghua.edu.cn chyyuu@gmail.com

2011年6月25日

报告内容

- 前言
- 国内外现状
- 实验课程设计
- 效果和存在的问题
- 小结

前言

- 对操作系统实验教学的理解
  - 计算机科学与计算机工程相结合
  - 原理和实验教学内容并行进行
  - 原理-->实验-->原理
  - 强调动手编程实践

前言

- 对操作系统实验教学的理解
  - 实验需求
    - 理解硬件
    - 循序渐进
    - 阅读代码
    - 把握全局
    - 功能完善
    - 改进创新

国外现状

- MIT：xv6 和JOS
  - 7千行以下，C语言，支持X86 SMP架构
- Harvard：OS161-1.4.1
  - 1万1千行代码，C语言，支持MIPS架构
- Columbia：Linux
  - 部分Linux核心代码，C语言
- Berkeley: Nachos
  - 1万行左右，C++ / java语言，模拟MIPS架构
- Stanford：PintOS
  - 1万1千行代码，C语言，
- Univ. of Maryland: geek OS
  - <10000行代码，C语言，x86

国内现状

- ucore
  - **清华** 基于jos/xv6/OS161/linux，200~10000行 C语言，支持X86-32/X86-64/ARM
- xv6和JOS
  - 北大
- Linux
  - 国防科大、浙大、西邮、清华
- MINIX
  - 上海交大，南开
- Nachos
  - 南开，山大
- Solaris Windows WRK Wince RTEMS ..

已有现状

- 清华大学的OS课程
  - 操作系统原理：本科大三下，160人左右
    - 原理与实验
  - 操作系统实践：本科大四上，60人左右
    - 侧重创新型实践，Linux....
  - 高级操作系统：研究生课程，40人左右
    - 操作系统前沿+研究型实验，SOSP，OSDI, EuroSys...

实验课程设计

- 目标
  - 对原理知识的补充和完善
    - 讲课内容和实验内容同步
  - 让学生对操作系统设计有一个全局的理解
    - 操作系统要小巧且覆盖面全
  - 适合不同层次学生的需求
    - 存在高中低三类学生

实验课程设计

- 设计思路
  - 方便且利用理解细节
    - 大量采用开源软件
    - 实验环境：Windows/Linux
    - IDE工具：Eclipse
    - 源码阅读工具：Kscope
    - 源码文档自动生成工具：Doxygen
    - 编译环境：gcc，make，Binutils
    - 真实/虚拟运行环境：X86机器或QEMU
    - 调试工具：改进的QEMU＋GDB

实验课程设计

- 设计思路
  - 采用小巧全面的操作系统ucore并进行改进，需要覆盖操作系统的关键点，为此增加：
    - I/O管理/中断管理
    - 虚存管理/页表/缺页处理/页替换算法
    - 进程管理/调度器算法
    - 信号量实现和同步互斥应用
    - 基于链表/FAT的文件系统
  - 完整代码量控制在10000行左右
  - 提供实验讲义和源码分析文档

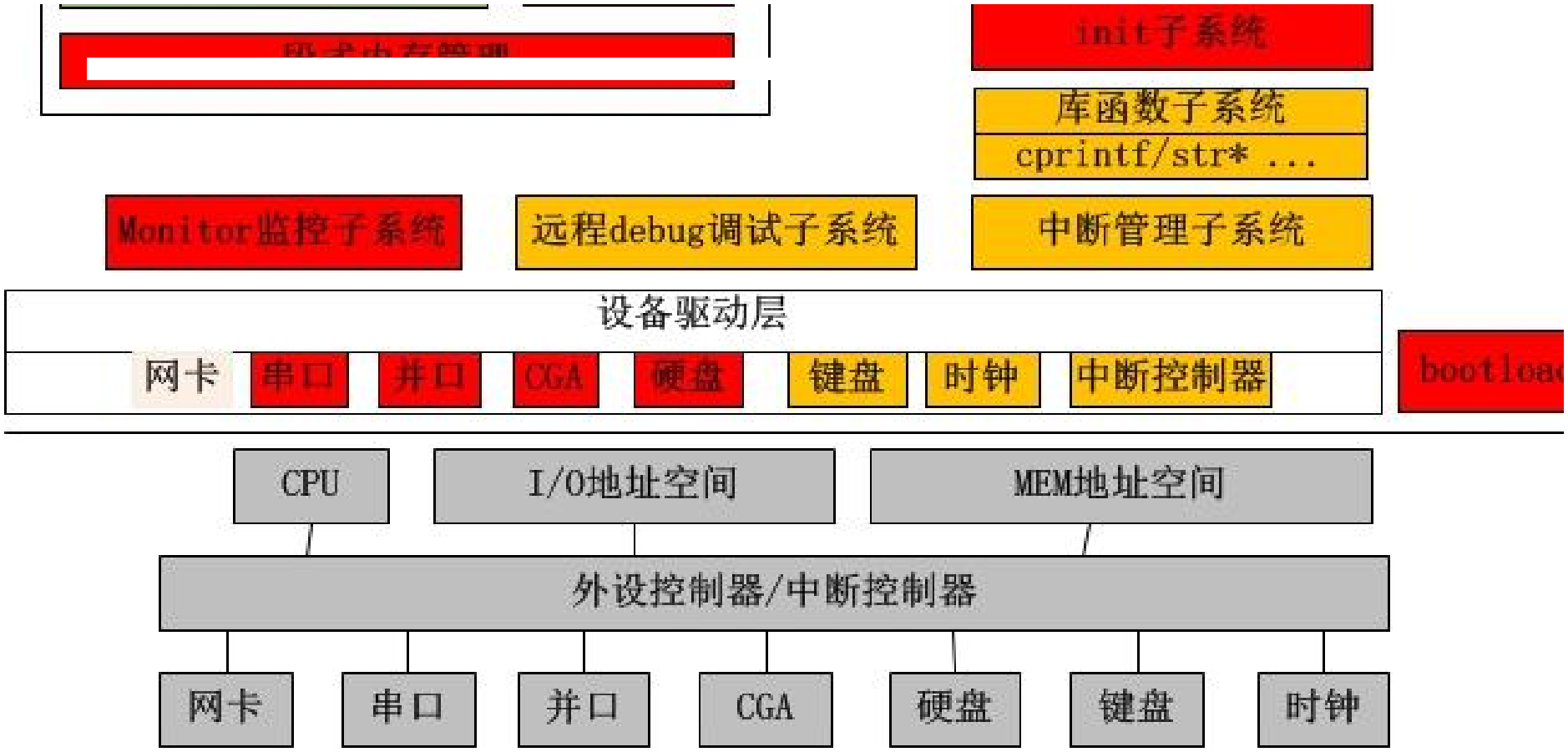
实验课程设计

- 设计思路
  - 利用互联网进行广泛的交流：
    - 建立邮件列表：答疑、交流、总结
    - 代码和实验讲义公开
    - 开发论坛： https://groups.google.com/group/ucore_dev
    - 实验使用论坛：https://groups.google.com/group/oscourse
    - 源码：http://code.google.com/p/ucore/
  - 差异化教学
    - 很好的学生：鼓励创新
    - 中差学生：鼓励根据各自层次选择合适的实验方式

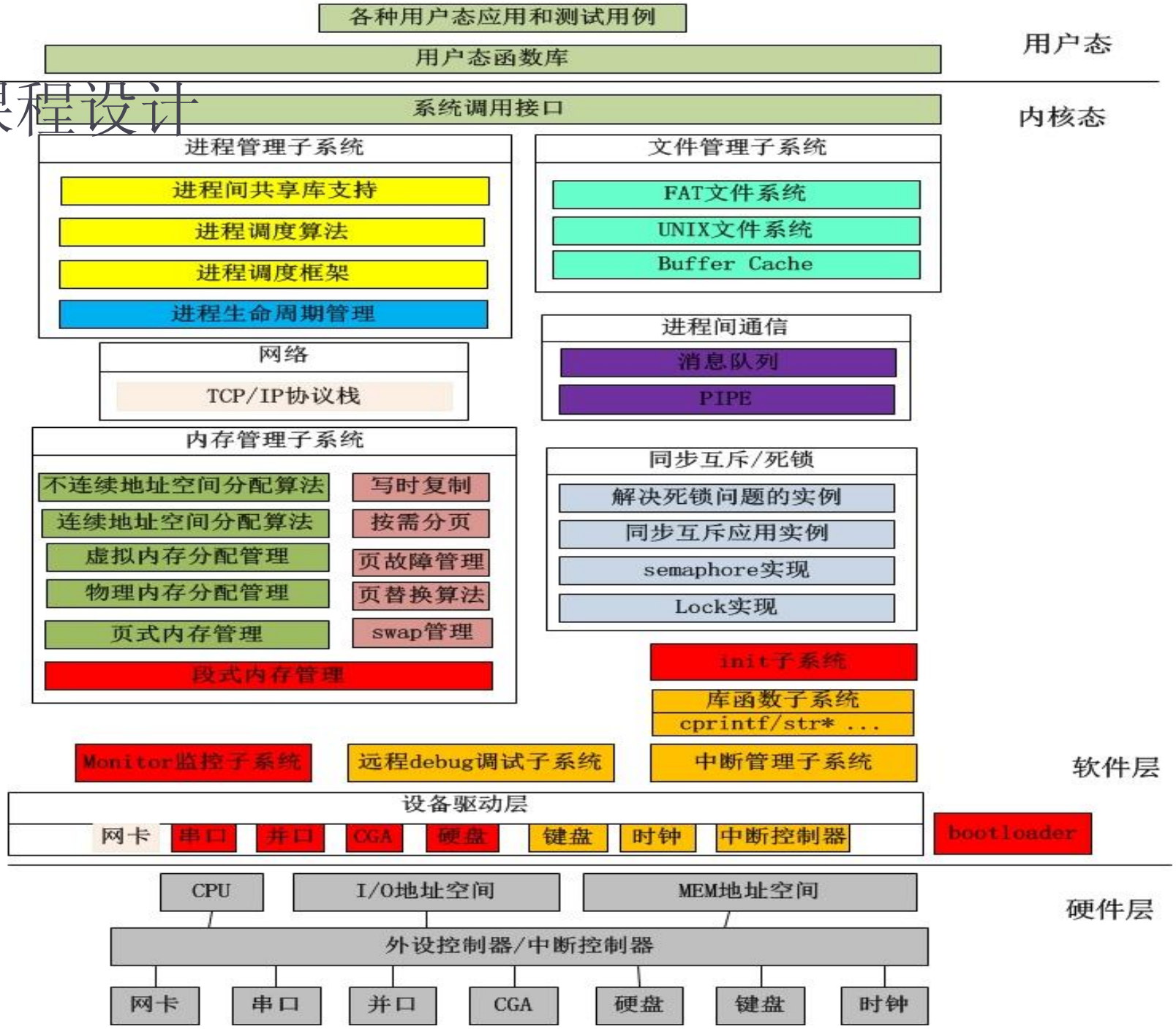
实验课程设计

- 实验内容
  - 1 OS启动、中断与设备管理：200~1617行
  - 2 内存管理：1800~3000行
  - 3 进程管理：3500~4100行
  - 4 处理器调度：4300~5000行
  - 5 同步互斥与死锁：5100~6400行
  - 6 文件系统：7500~9900行
  - 7 TCP/IP/net driver：10000~13000行

实验课程设计


init子系统
库函数子系统
cprintf/str* ...
Monitor监控子系统
远程debug调试子系统
中断管理子系统
设备驱动层
网卡
串口
并口
CGA
硬盘
键盘
时钟
中断控制器
CPU
I/O地址空间
MEM地址空间
外设控制器/中断控制器
网卡
串口
并口
CGA
硬盘
键盘
时钟

实验课程设计


各种用户态应用和测试用例
用户态函数库
用户态
系统调用接口
内核态
进程管理子系统
进程间共享库支持
进程调度算法
进程调度框架
进程生命周期管理
文件管理子系统
FAT文件系统
UNIX文件系统
Buffer Cache
网络
TCP/IP协议栈
进程间通信
消息队列
PIPE
内存管理子系统
不连续地址空间分配算法
连续地址空间分配算法
虚拟内存分配管理
物理内存分配管理
页式内存管理
段式内存管理
写时复制
按需分页
页故障管理
页替换算法
swap管理
同步互斥/死锁
解决死锁问题的实例
同步互斥应用实例
semaphore实现
Lock实现
init子系统
库函数子系统
cprintf/str* ...
Monitor监控子系统
远程debug调试子系统
中断管理子系统
软件层
设备驱动层
网卡
串口
并口
CGA
硬盘
键盘
时钟
中断控制器
bootloader
CPU
I/O地址空间
MEM地址空间
硬件层
外设控制器/中断控制器
网卡
串口
并口
CGA
硬盘
键盘
时钟

实验课程设计

- Lab1:
  - Bootloader/Interrupt/Device Driver
  - 7: proj1~4.1.1

- 基于分段机制的存储管理
- 设备管理的基本概念
- PC启动bootloader的过程
- bootloader的文件组成
- 编译运行bootloader的过程
- 调试bootloader的方法
- 在汇编级了解栈的结构和处理过程
- 中断处理机制
- 通过串口/并口/CGA输出字符的方法

```
proj1 /
|-- boot
|   |-- asm.h
|   |-- bootasm.S
|   `-- bootmain.c
|-- libs
|   |-- types.h
|   `-- x86.h
|-- Makefile
`-- tools
    |-- function.mk
    `-- sign.c
```

3 directories, 8 files

实验课程设计

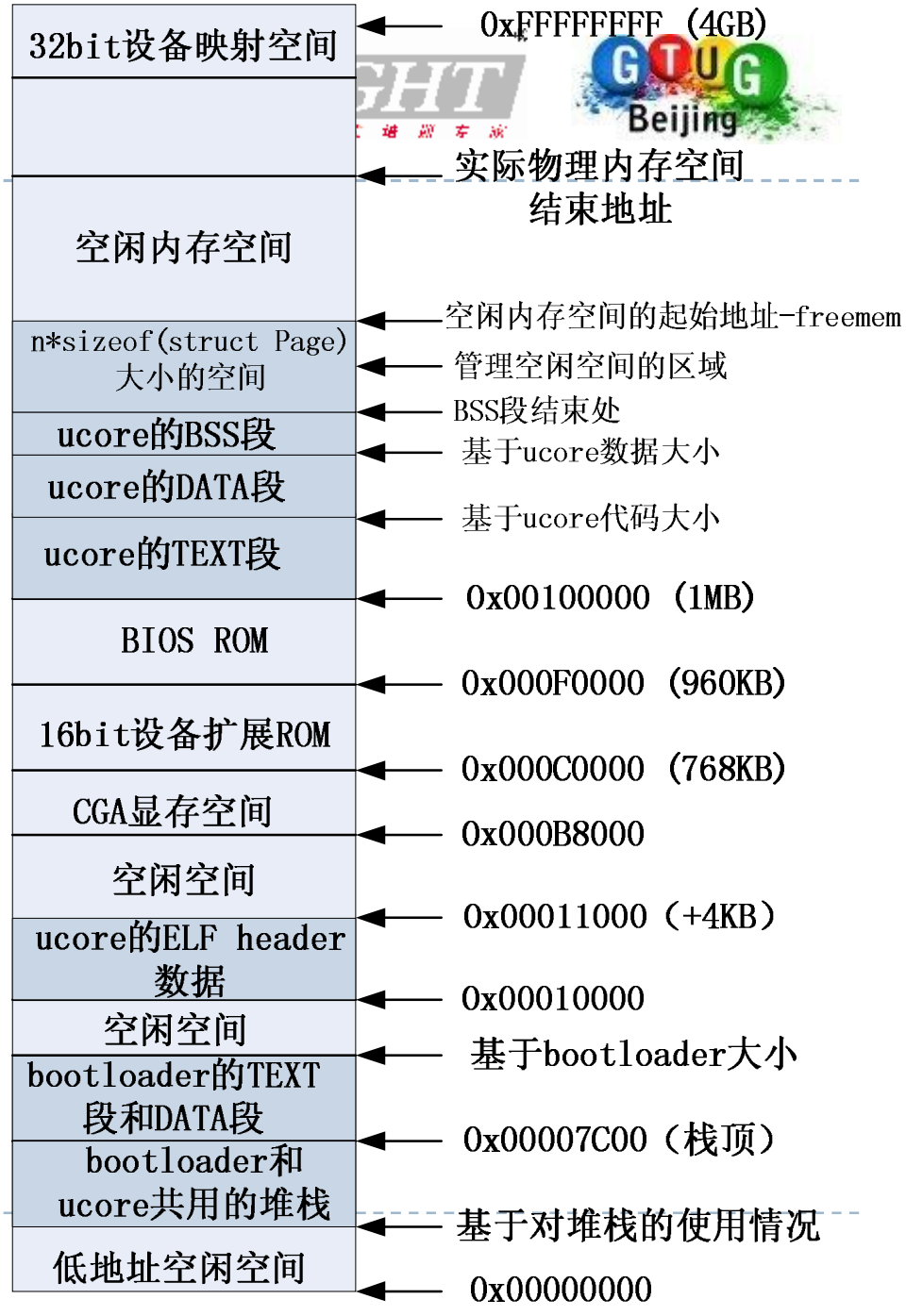
- Lab2
  - Memory management
  - 11：proj5~9.2

- 理解内存地址的转换和保护
- 理解页表的建立和使用方法
- 了解物理内存的管理方法
- 了解常用的减少碎片的方法
- 了解虚拟内存的管理方法

```
proj5
  |-- boot
  |   |-- asm.h
  |   |-- bootasm.S
  |   `-- bootmain.c
  |-- kern
  |   |-- init
  |   |   |-- entry.S
  |   |   `-- init.c
  |   |-- mm
  |   |   |-- default_pmm.c
  |   |   |-- default_pmm.h
  |   |   |-- memlayout.h
  |   |   |-- mmu.h
  |   |   |-- pmm.c
  |   |   `-- pmm.h
  |   |-- sync
  |   |   `-- sync.h
  |   `-- trap
  |       |-- trap.c
  |       |-- trapentry.S
  |       |-- trap.h
  |       `-- vectors.S
  |-- libs
  |   |-- atomic.h
  |   |-- list.h
  `-- tools
      |-- kernel.ld
```

实验课程设计

- Lab2
  - Memory management
  - 11：proj5~9.2
- 特点：
  - 覆盖OS知识点
  - 实现内存管理框架，便于扩展实现不同的内存分配算法
  - 代码量小



实验课程设计

Lab2：功能对比

功能	ucore	Linux
内存管理	段页式	段页式
物理页管理	Best/worst/first Fit、Buddy system	Buddy System
虚拟内存管理	基于局域的页替换/基于全局的页替换实现（参考Linux-2.4）	基于全局的页替换实现
Swap机制	实现	实现
任意大小的Slab分配	实现	实现, slob/slub
缺页异常处理	实现	实现
Copy On Write	实现	实现
Demanding Page	实现	实现

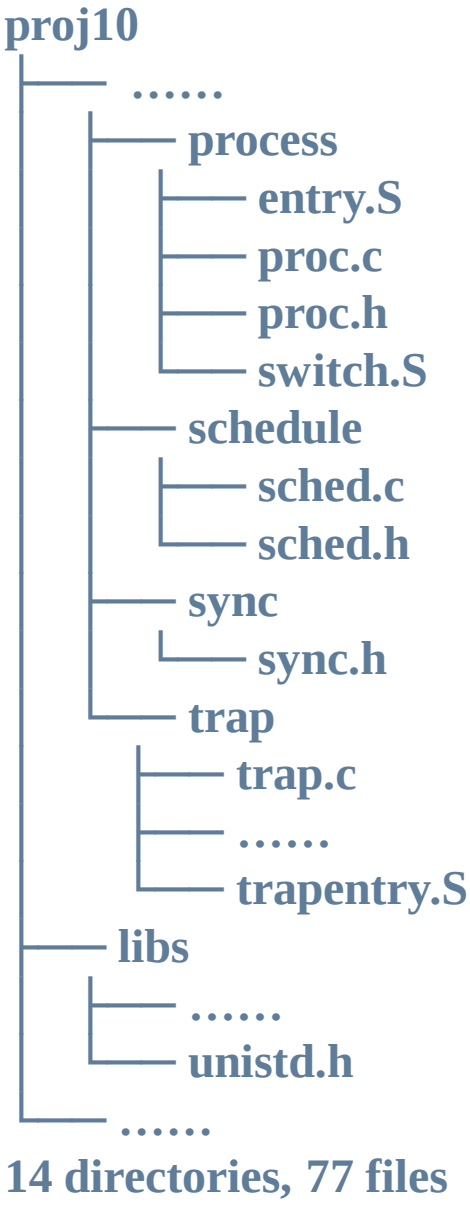
实验课程设计

- Lab3
  - Process management
  - 7：proj10~12

- 了解用户进程的关键信息
- 理解内核管理用户进程的方法
- 理解系统调用的过程

特点：

- 覆盖OS知识点
- 实现内核线程、用户进程/线程
- 实现系统调用
- 代码量小



实验课程设计

- Lab3：功能对比

功能	ucore	Linux
内核线程	实现	实现
用户进程	实现	实现
用户线程	实现	实现
Swap内核线程	实现	实现
基于COW的进程创建	实现	实现
时间机制	实现	实现
等待队列	实现	实现
进程/线程上下文切换	实现	实现

实验课程设计

- Lab4
  - Process scheduling
  - 3：proj13~13.2

- ·熟悉 ucore 的系统调度器框架，以及内置的 Round-Robin 调度算法。
- ·基于调度器框架实现一个调度器算法

特点：

- 覆盖OS知识点
- 实现调度框架，用于在同一的框架下设计不同的调度算法
- 代码量小


```
proj13
├── kern
│   ├── ......
│   ├── process
│   │   ├── ......
│   │   ├── proc.h
│   │   └── proc.c
│   ├── schedule
│   │   ├── sched.c
│   │   ├── sched_FCFS.c
│   │   ├── sched_FCFS.h
│   │   └── sched.h
│   └── trap
│       ├── trap.c
│       └── ......
└── user
    ├── matrix.c
    └── ......

17 directories, 129 files
```

实验课程设计

Lab4：功能对比

功能	ucore	Linux
调度框架	实现	实现
SMP支持	N/A	实现
CFS调度器	N/A	实现
Stride调度器	实现	N/A
FIFO/RR/MLFQ	实现	N/A
RT调度	N/A	实现
O(1)	N/A	实现

实验课程设计

- Lab5
  - Process scheduling
  - 5：proj14~16

- 熟悉 ucore 的同步互斥机制。
- 实现管程的条件变量

特点：

- 覆盖OS知识点
- 包含基本的spinlock、semphpore、condition variable
- 代码量小

```
Lab5
|—— lab5.pdf
|—— lab5-handin
`—— proj16
|—— ...
|—— libs
|                    |—— ...
|                    |—— atomic.h
|                    |—— types.h
|                    |—— unistd.h
|                    `—— list.h
|—— kern
|                    |—— ...
|                    `—— sync
|                    |         |—— sync.h
|                    |         |—— sync.c
|                    |         |—— wait.h
|                    |         |—— wait.c
|                    |         |—— sem.h
|                    |         |—— sem.c
|                    |         |—— condition.h
|                    |         `—— klock.c
|                    `—— syscall
|                              `——syscall.c
`—— user
|—— ...
                     `——libs
                     |         |——syscall.h
                     |         |——syscall.c
                     |         |——ulib.h
                     |         `——ulib.c
|—— semtest.c
```

实验课程设计

- Lab5：功能对比

功能	ucore	Linux
Atomic op	实现	实现
spinlock	实现	实现
ticketlock	N/A	实现
semaphore	实现	实现
RCU	N/A	实现
RWlock	N/A	实现
RW semaphore	N/A	实现
Condition variable	实现	N/A

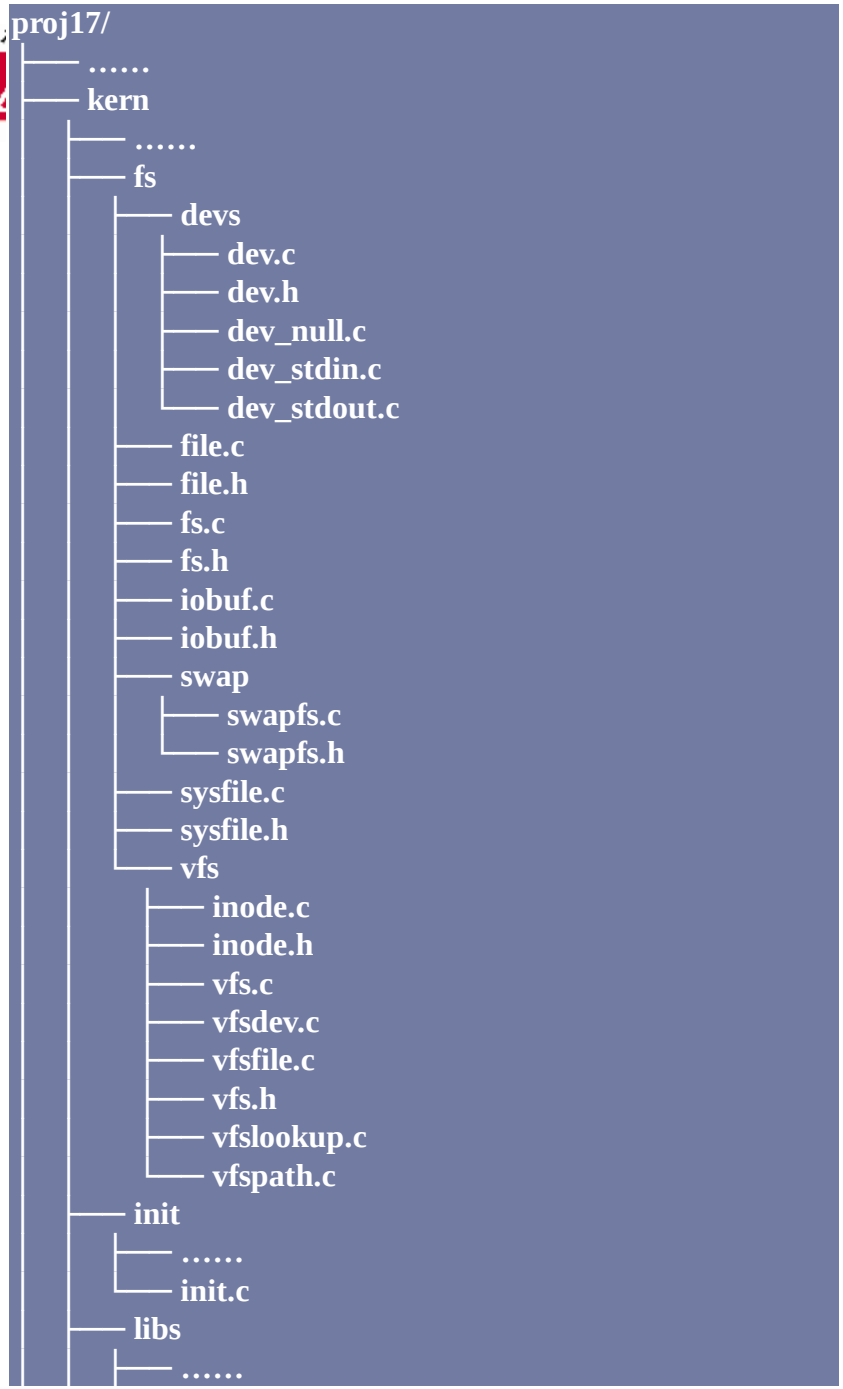
实验课程设计

- Lab6
  - File System
  - 3：proj13~13.2

- 掌握基本的文件系统系统调用的实现方法；
- 了解一个基于索引节点组织方式的Simple FS文件系统的设计与实现；
- 了解文件系统抽象层-VFS的设计与实现；

特点：

- 覆盖OS知识点
- 实现VFS
- 实现简化的Unix FS，FAT FS
- 代码量小



实验课程设计

Lab6：FS框架

实验课程设计

- Lab6：功能对比

功能	ucore	Linux
VFS	实现	实现
Device FS	实现	实现
UNIX FS	实现	实现
FAT FS	实现	实现
Ext2/…	N/A	实现

实验课程设计

扩展实验

U0: ucore porting on x86-64
Status: 100%, Done
ucorer: wnz

U1: local page replacement framework with different algorithms of local page replacement
status: 100%, Done
ucorer: yxh

U2: ucore支持ARM CPU(with mmu)，能够ucore for ARM在SkyEye（模拟S3C2410）或Android提供的qemu上运行起来。http://code.google.com/p/u2proj
Status: 70%, Done: lab1(intrrupt)/lab2(phy mem/page table/swap/page fault) Done.
Working: lab3(process)
ucorer: wjf, ykl, xb

U3：condition variable(used in monitor concept) implementation
status: 100%, Done
ucorer:lrl

实验课程设计

扩展实验

U5: ucore网络支持：支持TCP/IP协议栈(基于lwip)和无线有线网卡驱动
status: 90%, Done: lwip stack in ucore, network related function/syscall in bionic libc, E100 NIC driver
Working: E1000 NIC driver, socketpair syscall
ucorer: hg,zwl

U6: ucore SMP支持&内存置换算法：扩展ucore能够支持x86（32位）的多处理机，需要把现有的ucore的内存管理/进程管理部分尽量支持并行处理，并支持和实现多种新内存置换算法（要考虑并行处理）
http://code.google.com/p/u6proj
status: 0%
ucorer: ???

U9: ucore文件系统框架：支持在VFS下同时支持FAT32等文件系统，实现更加简化的VFS、FAT和SFS，即SVFS、SFAT&SSFS文件系统，并能够实现高性能的基于DMA方式的磁盘访问；http://code.google.com/p/u9proj
status: 90%, Done: FAT FS, Simple FS, VFS, DMA block driver， reduced FSes(VFS/FAT/SFS)
Working: Buffer cache
ucorer: qz,rsw

扩展实验

U10:ucore支持android for x86的Bionic Libc，从而进一步支持Dalvik JVM在其上运行.
http://code.google.com/p/u10proj1
status: 90%, Done: processs/memory/signal related syscall
Working: dynamic library support,other syscalls
ucorer: zc,st
U12:ucore支持GO programming language for
x86和对应runtime库（主要工作在ucore支持runtime库上，涉及对新型线程的支持和基于Garbage
Collection的内存管理支持） http://code.google.com/p/u12proj
status: 95%, Done: all but signal related runtime testsuits can run on ucore
Working: use U10 singal support in ucore and pass all GO testsuits.
ucorer: cr,fjy

U13:UMucore，修改ucore，把uocre实现成一个用户态的应用程序--User Mode Ucore，基本思路类似User Mode
Linux和status: 95%, Done: all but Debug support
Working: UMucore can be debugged by gdb
ucorer: mjj

效果和存在的问题

- 好的方面
  - 理论和实验能够较好地结合起来，不再感到OS课是一个只要死记硬背的课程了
  - 理解了一个OS的全局设计于实现，而不是一个一个分离的知识点
  - 掌握了许多OS原理上没有涉及或涉及不够的东西，比如中断/系统调用的实现，X86的段页机制，进程上下文如何切换的，内核态和用户态的具体区别是什么
  - 这是大学期碰到的最复杂的软件，学习了分析和设计大型系统软件的方法

效果和存在的问题

- 存在的问题
  - 对Linux和相关软件和X86保护模式中断等不熟悉
    - 上课讲解相关内容，提高学生分析代码和编程能力
  - 大三下课程繁重，在OS实验上花费时间多，对其他课程有一定的影响
    - 只要求完成部分实验，其它实验提供参考答案来理解
  - 对抄袭没有有效的监管手段
    - 实验成绩30%，考试成绩(包含原理和实验内容)70%
  - 好学生吃不饱，差学生感觉是“下地狱”
    - 实验要求和参考答案一起给
    - 进一步完善实验文档、辅助参考文档和辅助工具
    - 让3%的好学生参加OS科研，让差学生理解参考答案

小节

- 个人体会
  - 一个可实际运行的，代码基小且实验点覆盖操作系统各个关键知识点的微型实验OS平台对学生学好操作系统课程有极大的促进作用
  - 吸取国内外先进经验并进行再加工
  - 广泛使用丰富的开源软件
  - 教师讲课内容与实验内容紧密结合
  - 学生需要进行差异化实验教学
  - 建立开放的实验交流环境，实现知识复用

贡献者：

Wang Nai Zheng, Han Wentao, Zhang kaichen,Guo Xiaolin,Xue Tianfan,Hu gang,Liu Cao,Su Yu,Yuan Xinhao, Yang Jian, Cao Zhen, 2011届做大实验的同学.....

欢迎访问、下载使用相关文档和实验软件并加入OS Course讨论组进行交流！

开发论坛： https://groups.google.com/group/ucore_dev
实验使用论坛：https://groups.google.com/group/oscourse
源码：http://code.google.com/p/ucore/