操作系统实验探索
——以教学内核AIM为例

复旦大学计算机科学技术学院
张亮 高建
lzhang@fudan.edu.cn
2016-12-17

相关工作

- 常见的教学操作系统内核(xv6/JOS, μCore，OS/161, Nachos等)
  - 部分系统特性落后
    - xv6无虚拟内存交换支持、μCore各个分支间有一定程度的特性差异
  - 部分系统无完整参考实现
  - 大多数仅支持单一平台
    - OS/161、Nachos仅支持MIPS的模拟器，其它只支持i386/QEMU
  - 开发和开展实验比较繁琐

提纲

- AIM研发动机
- AIM设计思路与主要特点
- AIM操作系统实验设计
- AIM教学实践

AIM开发动机

- 服务于操作系统课程教学
  - AIM: 一个构件化的跨平台教学操作系统内核
    - https://github.com/davidgao/AIMv6 (内核上游版本)
    - https://github.com/davidgao/AIM-public (实验设计)
- 对比相关工作，体现现代操作系统原理和实现技术
  - 例如：虚存、多核支持、内核多线程、动态设备管理等
- 培养学生的系统能力
  - 既知树木，也见森林
- 鼓励学生动手实践(Learning-by-Doing)

AIM设计思路

- 精简易理解(构件化 + 控制规模)
- 体现现代操作系统的基本特性
- 逐级分离解耦
  - 完全无关→ISA相关→机器相关→设备相关→配置相关
- 定义内核标准接口，将具体实现以构件的形式接入
  - 妥善设计接口，减少编写、移植构件时的编码工作量
  - 削弱构件之间的耦合，允许替换构件
- 利用版本管理，简化开发、实施和维护工作

AIM主要特点

- 跨平台：同一Codebase支持多种指令集结构（ISA）
  - 目前ARM＋IA32＋MIPS
- 构件化：统一接口，支持不同的实现和系统配置
  - 内核中分配器和调度器都按构件化设计
  - 所有的驱动都按构件化设计
  - 基于Autotools的构件装配
- 运行于开发版或仿真器
  - Arm：Xilinx ZedBoard，ZYBO；Qemu
  - Ia32：Qemu（含kvm）
  - Mips：龙芯实验箱；Msim
- 现代操作系统特征
  - 虚存、多核支持、内核多线程、动态设备管理等

AIM的跨平台和构件替换

- 在configure时指定平台、参数和构件
- initcall机制
  - 构件可以完全模块化，向内核注册后完全通过回调工作，不触及内核的符号表，避免冲突
- 内核中的抽象层
  - early_console_init()等符号表调用
  - struct driver等统一接口的回调
  - 平台支持代码可以向内核注册回调入口，实现更复杂的交互，但不破坏抽象层之间的界限

系统演示：模块化及initcall


```
include/aim/initcalls.h:#define INITCALL_ROOTFS(fn)     INITCALL(fn, 6)
include/aim/initcalls.h:#define INITCALL_DEV(fn)        INITCALL(fn, 7)
include/aim/initcalls.h:#define INITCALL_SYSCALLS(fn)   INITCALL_SUBSYS(fn)
include/aim/initcalls.h:#define INITCALL_SCHED(fn)      INITCALL_SUBSYS(fn)
include/aim/initcalls.h:#define INITCALL_DRIVER(fn)     INITCALL_SUBSYS(fn)
include/syscall.h:      INITCALL_SYSCALLS(__##entry## init);
drivers/pci/sata_ide_pci.c:INITCALL_DRIVER(__pci_driver_init);
drivers/pci/pci.c:INITCALL_DRIVER(__driver_init);
drivers/ata/ata_ide.c:INITCALL_DRIVER(__driver_init);
drivers/serial/uart-zynq.c:INITCALL_DRIVER(__init)
drivers/serial/uart-msim-kernel.c:INITCALL_DRIVER(__driver_init);
drivers/serial/uart-ns16550-kernel.c:INITCALL_DRIVER(__driver_init);
drivers/block/msim-ddisk-kernel.c:INITCALL_DRIVER(__driver_init);
drivers/hd/dos.c:INITCALL_DRIVER(__dos_partition_table_init);
drivers/tty/tty.c:INITCALL_DRIVER(__driver_init);
drivers/io/io-port.c:INITCALL_DRIVER(__driver_init);
drivers/io/io-mem.c:INITCALL_DRIVER(__driver_init);
drivers/io/bus-mapper.c:INITCALL_DRIVER(__init);
drivers/sd/sd-zynq.c:INITCALL_DRIVER(__init)
fs/vnode.c:INITCALL_FS(vnodeinit);
fs/bio.c:INITCALL_FS(binit);
fs/spec.c:INITCALL_FS(specinit);
fs/mount.c:INITCALL_FS(mountinit);
fs/ufs/ext2fs/ext2fs.c:INITCALL_FS(ext2fs_register);
fs/ufs/hash.c:INITCALL_FS(ufs_ihashinit);
kern/vmaim.s:INITCALL_FS(binit);
kern/vmaim.s:INITCALL_FS(specinit);
kern/vmaim.s:INITCALL_FS(vnodeinit);
kern/vmaim.s:INITCALL_FS(vnodeinit);
kern/vmaim.s:INITCALL_FS(mountinit);
kern/mm/vmm/slab.c:EARLY_INITCALL(__init)
kern/mm/kmmap/ff.c:EARLY_INITCALL(__init)
kern/mm/kmmap/kmlist.c:EARLY_INITCALL(__init)
kern/dev/index/devlist.c:INITCALL_CORE(__init)
kern/proc/sched/sched_plain.c:INITCALL_SCHED(__sched_plain_init);
[10:19:34] david@N75SL /home/david/git/AIMv6 [arm-dev] (0)
```

```
KERN: Applying kmmap to: pgindex=0x8003c000.
pgindex: 8003c000
creating uvm
destroying uvm1
destroying uvm2
destroying mm
another mm
KERN: Applying kmmap to: pgindex=0x8003c000.
pgindex: 8003c000
yet another mm
KERN: Applying kmmap to: pgindex=0x80040000.
new pgindex: 80040000
yet ... another mm
KERN: Applying kmmap to: pgindex=0x8003c000.
pgindex: 8003c000
==========mm_test() finished==========
KERN: Scheduler initialized.
DEBUG: initcalls 0x800185e0 to 0x80018658
DEBUG: executing 0x800030cc
KERN: <devlist> initializing.
KERN: <devlist> Done.
DEBUG: executing 0x800004b4
DEBUG: executing 0x80000540
DEBUG: executing 0x80000bd0
DEBUG: executing 0x80001260
DEBUG: executing 0x80001370
DEBUG: executing 0x80001480
DEBUG: executing 0x80001a44
DEBUG: executing 0x80001cec
DEBUG: executing 0x80001e84
DEBUG: executing 0x80002454
DEBUG: executing 0x80002434
DEBUG: executing 0x800028c0
DEBUG: executing 0x80002bf8
DEBUG: executing 0x80002e50
DEBUG: executing 0x80003010
DEBUG: executing 0x800030ac
DEBUG: executing 0x800042e0
DEBUG: executing 0x800045e8
KERN: <uart-zynq> Initializing.
KERN: <uart-zynq> Ready.
DEBUG: executing 0x800050c0
DEBUG: executing 0x800052a8
DEBUG: executing 0x8000544c
KERN: <sd-zynq> Initializing.
KERN: <sd-zynq> Ready.
DEBUG: executing 0x80006ed4
DEBUG: <bus-mapper> initializing.
DEBUG: <bus-mapper> done.
DEBUG: executing 0x80014694
DEBUG: <io-mem> initializing.
```

```
记录了0+1 的读入
记录了0+1 的写出
268 bytes copied, 0.0411955 s, 6.5 kB/s
N75SL:/home/david/git/AIMv6 # dd if=kern/vmaim.elf of=/dev/loop0p
loop0p1  loop0p2  loop0p3
N75SL:/home/david/git/AIMv6 # dd if=kern/vmaim.elf of=/dev/loop0p2
记录了1602+1 的读入
记录了1602+1 的写出
820424 bytes (820 kB, 801 KiB) copied, 0.0353688 s, 23.2 MB/s
N75SL:/home/david/git/AIMv6 # mount /dev/loop0p1 /mnt/fat/ && cp firmware/arch/armv7a/firmware
.bin /mnt/fat/ && umount /mnt/fat
N75SL:/home/david/git/AIMv6 #
```

系统演示：内核编译配置

TermInix: Default

1: fish /home/david/git/AIMv6

```
config.status: creating user/sbin/Makefile
config.status: creating user/sbin/init/Makefile
config.status: creating config.h
config.status: config.h is unchanged
config.status: executing depfiles commands
config.status: executing libtool commands
configure:
========================
  Configuration Summary
========================

General
--------
  version:      0.1
  build:        x86_64-suse-linux-gnu
  host:         arm-unknown-eabi
  compiler:     arm-unknown-eabi-gcc
  CPPFLAGS:
  CFLAGS:       -g
  LDFLAGS:

* Flags are overridden by architecture and machine-specific flags,
* further overridden by per-target flags,
* and further overridden by flags passed to `make`.

Target
--------
  architecture: armv7a
  machine:      zynq
  processors:   2/NR_CPUS
  memory:       512M

Kernel
--------
  DEBUG_OUTPUT: yes
  KERN_BASE:    0x80000000
  KMMAP_BASE:   0xf0000000
  RESERVED:     0xfffff000
  USERTOP:      0x7ff00000
  USTACKSIZE:   32768
  Timer freq:   100
  Kernel stack: 32768

Algorithms
--------
  simple_allocator:     FLFF
  page_allocator:       FF
```

3: david@N75SL:/home/david/git/AIMv6

```
.bin /mnt/fat/ && umount /mnt/fat
N75SL:/home/david/git/AIMv6 #
```

2: fish /home/david/git/AIMv6

```
KERN: Applying kmmap to: pgindex=0x8003c000.
pgindex: 8003c000
creating uvm
destroying uvm1
destroying uvm2
destroying mm
another mm
KERN: Applying kmmap to: pgindex=0x8003c000.
pgindex: 8003c000
yet another mm
KERN: Applying kmmap to: pgindex=0x80040000.
new pgindex: 80040000
yet ... another mm
KERN: Applying kmmap to: pgindex=0x8003c000.
pgindex: 8003c000
==========mm_test() finished==========
KERN: Scheduler initialized.
DEBUG: initcalls 0x800185e0 to 0x80018658
DEBUG: executing 0x800030cc
KERN: <devlist> initializing.
KERN: <devlist> Done.
DEBUG: executing 0x800004b4
DEBUG: executing 0x80000540
DEBUG: executing 0x80000bd0
DEBUG: executing 0x80001260
DEBUG: executing 0x80001370
DEBUG: executing 0x80001480
DEBUG: executing 0x80001a44
DEBUG: executing 0x80001cec
DEBUG: executing 0x80001e84
DEBUG: executing 0x80002454
DEBUG: executing 0x80002434
DEBUG: executing 0x800028c0
DEBUG: executing 0x80002bf8
DEBUG: executing 0x80002e50
DEBUG: executing 0x80003010
DEBUG: executing 0x800030ac
DEBUG: executing 0x800042e0
DEBUG: executing 0x800045e8
KERN: <uart-zynq> Initializing.
KERN: <uart-zynq> Ready.
DEBUG: executing 0x800050c0
DEBUG: executing 0x800052a8
DEBUG: executing 0x8000544c
KERN: <sd-zynq> Initializing.
KERN: <sd-zynq> Ready.
DEBUG: executing 0x80006ed4
DEBUG: <bus-mapper> initializing.
DEBUG: <bus-mapper> done.
DEBUG: executing 0x80014694
DEBUG: <io-mem> initializing.
```

实验设计(OS AY2016)

- Lab0. 工具链准备: 平台相关 (第1周)
- Lab1. Booting AIM: 平台相关 (第2周)
- Lab2. 内存管理 I: 一定程度平台相关 (第3-5周)
- Lab3. 中断与异常: 平台相关 (第6周)
- Lab4. SMP:平台相关 (第7周)
- Lab5. 进程管理:平台无关 (第8-9周)
- Lab6. 设备驱动:设备相关 (第10-13周)
- Lab7. 系统调用: 平台无关 (第14周)
- Lab8. 内存管理 II: 平台无关 (第15周)

实验 X：OS挑战课题（AY2016）

AIM 是一个需要继续开发和维护的工程，我们将其中一部分任务设计为挑战课题。这些课题难度较大，并且没有参考实现，也没有固定的思路或者方法。选择并完成其中任何一个题目，可以展示你对操作系统的深入理解，以及强大的设计、实现能力，并得到实验成绩A。即使未能完成题目，你也可以依靠普通实验得到应有的评定，成绩不会受到影响。

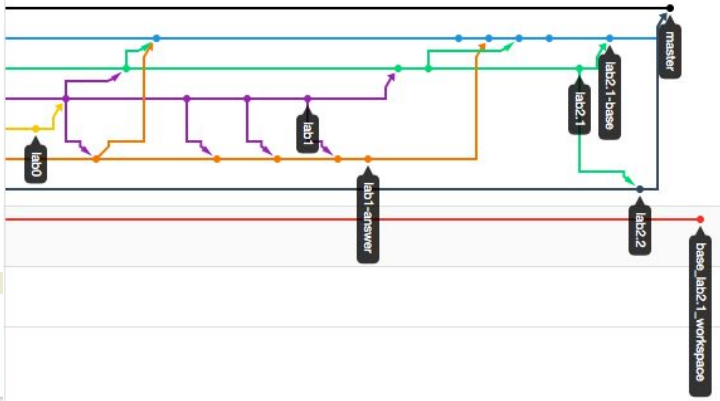
验收方式

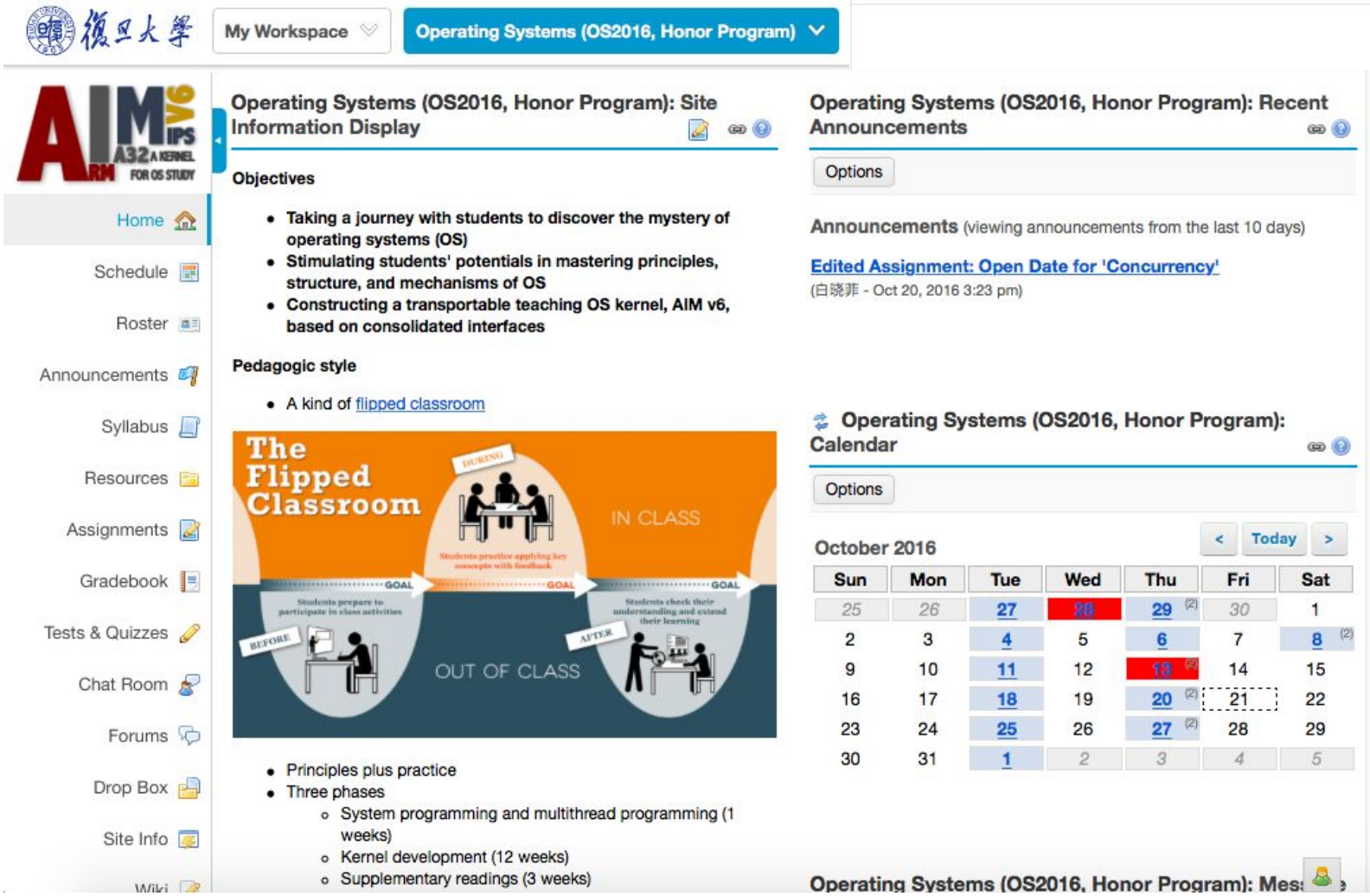
提交代码和报告（需要在学校留档），并在实验课上进行答辩。

题目（每人选择一题即可）

1. AIM 中有一小部分组件没有按照并发和并行运行的需求设计同步逻辑，另有部分组件的同步逻辑中存在较难重现的竞争条件，请对 AIM 的同步设计进行彻底的检查，找出并修复这些问题。
2. AIM 现有的代码包括有统一接口的页表操作和设备访问，各平台也能捕获内存访问异常并获取相关信息，请在保持跨平台和模块化设计的条件下，设计实现换页机制，包括设计内存访问异常的收集、处理和分发接口，以及设计简易的交换文件系统（也可以兼容现有交换文件系统）。
3. AIM 现有的代码实现了许多内核内部的功能，但是大部分功能并没有作为系统调用进行封装。请你参考 POSIX 标准，设计一套适合教学使用的系统调用接口，并按这套接口对 AIM 内部的功能进行封装，并实现线程库在用户态的组件。AIM 的系统调用是模块化的，很容易添加，由内核实现线程需要的功能也已基本实现好。

AIM教学实践与效果


master
lab2.1-base
lab2.1
lab1
lab0
lab1-answer
lab2.2
base_lab2.1_workspace


復旦大學
My Workspace
Operating Systems (OS2016, Honor Program)
AIM v6 IPS A32 A KERNEL FOR OS STUDY
Home
Schedule
Roster
Announcements
Syllabus
Resources
Assignments
Gradebook
Tests & Quizzes
Chat Room
Forums
Drop Box
Site Info
Operating Systems (OS2016, Honor Program): Site Information Display
Objectives
Taking a journey with students to discover the mystery of operating systems (OS)
Stimulating students' potentials in mastering principles, structure, and mechanisms of OS
Constructing a transportable teaching OS kernel, AIM v6, based on consolidated interfaces
Pedagogic style
A kind of flipped classroom
The Flipped Classroom
IN CLASS
OUT OF CLASS
BEFORE
AFTER
GOAL
Principles plus practice
Three phases
System programming and multithread programming (1 weeks)
Kernel development (12 weeks)
Supplementary readings (3 weeks)
Operating Systems (OS2016, Honor Program): Recent Announcements
Options
Announcements (viewing announcements from the last 10 days)
Edited Assignment: Open Date for 'Concurrency'
(白晓菲 - Oct 20, 2016 3:23 pm)
Operating Systems (OS2016, Honor Program): Calendar
Options
October 2016
Today
Sun Mon Tue Wed Thu Fri Sat
25 26 27 28 29 30 1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31 1 2 3 4 5

实验情况

- 拔尖班16名学生，大三为主
- 两人选择挑战项目，均已接近完成
- 大部分学生能跟上进度完成实验计划
  - 普遍情况，每周需要5-15小时
- 学生在课后进行了较多的讨论
  - 思路相似但各自实现
- 缺少必要基础知识的学生，跟进实验计划有很大难度

学生反馈

- 对操作系统有了更明确的认识
  - 覆盖从整体设计到实现细节的多个尺度
  - 与计算机原理、体系结构等课程衔接
- 能够在系统层面进行开发
  - 描述部件的行为
  - 设计实现符合行为要求的部件
- 养成了良好的开发习惯

总结

- 借助AIM构件化的设计特征
  - 帮助学生在按照系统观点理解操作系统内核
  - 帮助学生了解操作系统内核的跨平台问题
- 突出操作系统教学中的若干要点
  - 系统观(Systematic perspective)
  - 从实践中学习(Learning-by-Doing)
  - 设计与接口的分离(Separation of interface and implementation)
  - 力求简单(KISS of UNIX)
- 在真实的机器(开发板、实验箱、kvm)上开发教学操作系统内核

感谢

- 复旦大学计算机科学技术学院系统类课程建设的持续支持
- 教指委的指导和兄弟院校的交流
- 龙芯集团的平台支持
- 华章集团的鼓励和鞭策
- 开发团队的努力工作
- 您的关注