Lab4 实验介绍

操作系统实习课程

2008.4.16

主要内容

0011 0010 1010 1101 0001 0100 1011

- Lab4时间安排
- Lab4实习要求
- Lab4任务列表

Lab4时间安排

- Lab4时间：4月16日至4月29日（2周）
- 第1周
  - PartA　实现调度算法及创建新的environment
  - PartB　实现fork函数和用户态page fault
- 第2周
  - PartC　实现可剥夺的调度及IPC
  - 必做challenge

（lab4内容较多，请抓紧时间）

Lab4实习要求

0011 0010 1010 1101 0001 0100 1011

- Exercise1~11
  - 必做
- Questions
  - 必做，写入文档
- Challenges
  - 必做：1，2，5
  - 其他：选做

Lab4任务列表

- Round-robin调度算法
- 修改env结构的系统调用
- 实现Copy_on_Write的fork函数
- 实现用户态的page fault处理函数
- 在时钟中断中调用调度函数，实现可剥夺的调度
- 实现进程间消息通信

第一周任务清单

0011 0010 1010 1101 0001 0100 1011

Part A

实现一些系统调用函数：

1、round-robin调度算法：当进程自愿放弃CPU或者退出时调用，在Part C会实现时钟中断控制下的可剥夺调度

2、创建新的environment

第一周任务清单

0011 0010 1010 1101 0001 0100 1011

Exercise 1

- 填写sched_yield()函数，实现round-robin调度
- 不要忘记在syscall()中分发sys_yield()

第一周任务清单

0011 0010 1010 1101 0001 0100 1011

Exercise 2

实现kern/syscall.c中的下列函数：

- sys_exofork
- sys_env_set_status
- sys_page_alloc
- sys_page_map
- sys_page_unmap
- 可能会用到kern/pmap.c and kern/env.cc 中的若干函数，尤其是envid2env()。从现在开始，无论何时调用envid2env()，给checkperm 参数传递1。请注意检查无效的系统调用号，若无效则返回-E_INVAL。用user/dumbfork测试你的程序

第一周任务清单

0011 0010 1010 1101 0001 0100 1011

Part B

- 实现Copy_on_Write的fork函数
- 实现用户态的page fault处理函数

第一周任务清单

0011 0010 1010 1101 0001 0100 1011

Exercise 4

- 实现 sys_env_set_pgfault_upcall系统调用，注意当查找目标进程的进程号时进行权限检查

第一周任务清单

0011 0010 1010 1101 0001 0100 1011

Exercise 5

- 修改kern/trap.c，使得其把page faults 分发给用户态的处理函数。注意当写入exception stack时采用一定的安全检查，如栈空间不够

page_fault_handler(struct Trapframe *tf)函数是具体处理page fault的函数，其中Exception Stack的设置参考文档中画的图

第一周任务清单

0011 0010 1010 1101 0001 0100 1011

Exercise 6

- 实现lib/pfentry.S 中的_pgfault_upcall，该函数将直接返回到引起缺页的代码，而不必经过内核。难点是同时变换堆栈和EIP
- 该函数是用户态page fault处理函数的总入口，负责调用具体的page fault处理函数，并返回到引起page fault的代码处执行
- 该函数由下一个待实现的代码注册到进程的env结构中

第一周任务清单

0011 0010 1010 1101 0001 0100 1011

Exercise 7

- 完成lib/pgfault.c 中的set_pgfault_handler()

第一周任务清单

0011 0010 1010 1101 0001 0100 1011

Exercise 8

- 实现lib/fork.c 中的fork 和 pgfault

- 这里的fork是创建新进程的总入口
- pgfault是具体的page fault处理函数，根据情况分配新页或者进行其他工作

第二周任务清单

0011 0010 1010 1101 0001 0100 1011

Part C

- 在时钟中断中调用调度函数，实现可剥夺的调度
- 实现进程间消息通信

第二周任务清单

0011 0010 1010 1101 0001 0100 1011

Exercise 9

- 修改 kern/trapentry.S 和 kern/trap.c 来初始化 IDT中的相应项，为 IRQs 0到 15提供处理函数，然后修改 kern/env.c 中的env_alloc() 来保证用户态下运行时中断允许

第二周任务清单

0011 0010 1010 1101 0001 0100 1011

Exercise 10

- 修改 trap_dispatch()函数，这样当时钟中断发生时，它调用 sched_yield()来寻找和运行另外一个进程

第二周任务清单

0011 0010 1010 1101 0001 0100 1011

Exercise 11

- 实现kern/syscall.c 中的sys_ipc_recv 和 sys_ipc_can_send。在以上函数中当调用 envid2env 时设置 checkperm 为 0，这样任何进程都可以发送消息给任何其他进程，内核仅仅检查目标进程是否存在

第二周任务清单

0011 0010 1010 1101 0001 0100 1011

- Challenge 1: 必做
  - 实现有优先级的调度算法，及相关的优先级存储、设置等功能
- Challenge 2: 必做
  - 内核为用户程序保护其使用的FPU、SSE等扩展寄存器数据，避免多个使用这些寄存器的用户进程互相破坏数据
  - 实现PCB中相关的数据存储代码，最好能让使用扩展寄存器的用户进程申请后才替其保护

第二周任务清单

- Challenge 3: 选做
  - 实现对用户进程信息进行快照和重放的系统调用，使得父进程可以定时监控子进程的活动，并尝试将子进程恢复到某个状态
  - 需要自己考虑，要达到进程状态重放需要保留那些数据？PCB？内存？堆栈？
- Challenge 4: 选做
  - 允许用户态进程设置自己处理大部分中断、异常

第二周任务清单

0011 0010 1010 1101 0001 0100 1011

- Challenge 5: 必做
  - 实现不使用COW，而使用共享内存的sfork()
- Challenge 6: 选做
  - fork()中使用大量的系统调用在ring0和ring3间切换开销很大，如何更改系统调用的接口来减少这些开销？
- Challenge 7: 选做
  - 实现公平的ipc_send()函数并测试

第二周任务清单

0011 0010 1010 1101 0001 0100 1011

- Challenge 8: 选做
  - 修改内核和用户态lib的实现，使ipc_send()中不需要循环等待
- Challenge 9: 选做
  - 实现参考文献中更优的素数计算方法
- Challenge 10: 选做
  - 实现参考文献中的正弦算法

第二周任务清单

0011 0010 1010 1101 0001 0100 1011

- Challenge 11: 选做
  - 使用参考文献中的方法或自己的方法，优化IPC调用
- Challenge 12: 选做
  - 当前IPC只能传递int或一个页面，实现能传递更复杂内容的IPC

END