中国科学技术大学
University of Science and Technology of China

操作系统 实验二

Nachos进程管理与调度

（一个简单shell的实现）

任课教师：李永坤

助教：田成锦、李佳伟

2018.04.16

Nachos进程管理与调度

- 实验目的
  - 掌握进程管理与同步：实现fork、exec、join系统调用.
  - 掌握进程调度：实现优先级调度.
- 实验内容
  - 实现fork、exec、join系统调用.（进程管理、同步）
  - 实现进程优先级调度.（进程调度）
  - 编写一个简单的Nachos shell.（具体运用）

实验说明

- 理论
  - 理论课上的fork、exec、waitpid / join 工作原理.
- 实现
  - 在Nachos中实现fork、exec、join系统调用.
- 运用
  - 运用fork、exec、join编写一个简单的Nachos shell解释器.
- 扩展
  - 当shell中输入多条指令时，内核能实现优先级调度.

中国科学技术大学
University of Science and Technology of China

1、实现fork、exec、join系统调用

2、实现简单的Nachos shell

3、Nachos进程优先级调度

中国科学技术大学
University of Science and Technology of China

1、实现fork、exec、join系统调用

2、实现简单的Nachos shell

3、Nachos进程优先级调度

1. 实现fork、exec、join

- fork
  - 原型：int Fork ();
  - 功能：创建一个子进程（父子进程运行相同的代码）
  - 返回值：父进程返回子进程id，子进程返回0
- exec
  - 原型：void Exec (char *exec_name);
  - 功能：载入并运行其他程序
  - 参数： exec_name：要运行的可执行程序的文件名

1. 实现fork、exec、join

（续1）

- join
  - 原型：void Join (int child_id);
  - 功能：父进程等待某个子进程执行结束
    （类似课上讲的waitpid函数）
  - 参数： child_id ： 子进程的id

1. 实现fork、exec、join

中国科学技术大学
University of Science and Technology of China

（续2）

- 如何测试？
  - fork的测试
    - ✓ 测试程序：test目录下的fork.c
  - exec的测试
    - ✓ 测试程序：test目录下的exec.c
  - join的测试
    - ✓ 测试程序：test目录下的join.c
- 验收标准
  - 能正确运行上述三个测试

中国科学技术大学
University of Science and Technology of China

1、实现fork、exec、join系统调用

2、实现简单的Nachos shell

3、Nachos进程优先级调度

2. 实现Nachos shell

- 什么是shell？

  – shell是用户和操作系统之间的接口.

  ✓ 如Windows的PowerShell、Linux的Bash Shell

  ✓ 用户可通过shell执行操作系统的指令

  - 如Linux shell中执行ls、pwd、cat等指令

2. 实现Nachos shell（续1）

- 实验要求
  - 补全test/shell.c程序，要求实现如下功能：
    - ✓ 运行可执行文件.（如./add）
      - ~~./add的含义是运行当前目录下的可执行程序add~~
    - ✓ 并行执行多条指令.（如./add ; ./sub ; ./join ; ./fork）
      - 指令间用英文分号隔开，且假设中间无空格
    - ✓ 指令运行结束后，回到shell，继续接收用户输入，循环执行
    - ✓ 必须使用多进程的方式实现
      - 父进程负责循环整个shell，有命令时交给子进程完成
      - 执行多条指令时，子进程数量 = 指令数量，且每个子进程负责一条指令
- 验收标准
  - 能正确执行单指令、多指令

中国科学技术大学
University of Science and Technology of China

1、实现fork、exec、join系统调用

2、实现简单的Nachos shell

3、Nachos进程优先级调度

3. 进程优先级调度

- 背景
  - 当shell同时运行多条指令时，希望优先执行高优先级指令
  - 进程调度决定当多进程处于就绪态时如何选择下一个运行的进程
  - Nachos目前只支持RR调度（轮流执行进程）

3. 进程优先级调度（续1）

- 实验要求

  - 优先级调度：静态（助教已实现）
    - ✓为进程增加优先级，高优先级进程先执行

  - 优先级调度：动态（需要你实践）
    - ✓在进程切换时动态调整进程优先级

3. 进程优先级调度（续2）

- 动态优先级调度的流程：
  - 创建进程时，为其赋予优先级
  - 进程调度切换上下文时：
    - ✓ 降低当前进程优先级， 增加所有就绪进程的优先级
    - ✓ 切换到当前优先级最高的进程执行
- 验收标准
  - 观察调度结果，能看到每次调度都是高优先级进程先执行

中国科学技术大学

University of Science and Technology of China

前期准备工作

前期准备

- 阅读相关类的代码
- 明确每个类函数的功能和实现流程

需要阅读的类有

- Thread类（ threads目录下thread.h、thread.cc ）
- AddrSpace类（ userprog目录下addrspace.h、 addrspace.cc ）
- Scheduler类（ threads目录下scheduler.h 、 scheduler .cc）
- Semaphore类（ threads目录下synch.h 、 synch .cc）
- Kernel类（ threads目录下kernel.h、kernel.cc ）
- （验收时的问题会涉及到这些类）

续言

- 本次实验形式为代码填空
  - 必须且只能把代码写在助教圈定的范围以内
  - 不能在任何地方添加任何头文件
  - 不能使用C或C++的库函数（如vfork, wait, exec函数族等）

```
//begin: 实验二的代码请写在这里++++++++++++++++++++++
// TODO:
//更新当前线程的优先级
//flushPriority();刷新所有就绪线程的优先级
flushPriority();
//end: ++++++++++++++++++++++++++++++++++++++++++++++

Print(); //打印当前进程状态。（助教已实现函数，这个函数不要修改
```

续言

- 本次分两阶段进行
  - 第一阶段实验内容为：实现fork、exec、join系统调用
  - 第二阶段实现所有内容

- 详细实验培训、验收标准参照:

  《os2018实验手册-lab2.pdf 》

The end~