操作系统实验2016

Lab 04

Contents

实验提交

截止时间: 2016/05/11 23:59:59 (如无特殊原因，迟交的作业将损失50%的成绩(即使迟了 1 秒)，请大家合理分配时间)

请大家在提交的实验报告中注明你的邮箱，方便我们及时给你一些反馈信息。

学术诚信: 如果你确实无法完成实验，你可以选择不提交，作为学术诚信的奖励，你将会获得10%的分数；但若发现抄袭现象，抄袭双方(或团体)在本次实验中得 0 分。

提交地址：http://cslabcms.nju.edu.cn/

提交格式: 你需要将整个工程打包上传，特别地，我们会清除中间结果重新编译，若编译不通过，你将损失相应的分数（请在报告中注明你实验所使用的 gcc 的版本，以便助教处理一些 gcc 版本带来的问题）. 我们会使用脚本进行批量解压缩. 压缩包的命名只能包含你的

学号。另外为了防止编码问题，压缩包中的所有文件都不要包含中文.如果你需要多次提交，请先手动删除旧的提交记录(提交网站允许下载，删除自己的提交记录)，否则若脚本解压时出现多次提交相互覆盖的现象，后果自负.我们只接受以下格式的压缩包：

- tar.gz
- tar.bz2
- zip

若提交的压缩包因格式原因无法被脚本识别，后果自负。

请你在实验截止前务必确认你提交的内容符合要求(格式、相关内容等)，你可以下载你提交的内容进行确认。如果由于你的原因给我们造成了不必要的麻烦，视情况而定，在本次实验中你将会被扣除一定的分数，最高可达 50% 。

git 版本控制：我们建议你使用 git 管理你的项目，如果你提交的实验中包含均匀合理的 git 记录，你将会获得 10% 的分数奖励（请注意，本实验的 Makefile 是由你自己准备的，你可以选择像 PA 中一样在每一次 make 后增加新的 git 记录作为备份，但是请注意，这样生成的 git log 一般是无意义的，所以不能作为加分项）。为此，请你确认提交的压缩包中包含一个名为 .git 的文件夹。

实验报告要求：仅接受 pdf 格式的实验报告，不超过 3 页 A4 纸，字号不能小于五号，尽可能表现出你实验过程的心得，你攻克的难题，你踩的不同寻常的坑。

分数分布： - 实验主体：80% - 实验报告：20%

解释：

1. 每次实验最多获得满分；
2. git 的分数奖励是在实验主体基础上计算的
3. git 记录是否“均匀合理”由助教判定；
4. 迟交扣除整个实验分数的 50% ；
5. 作弊扣除整个实验分数的 100% ；
6. 提交格式不合理扣除整个实验分数的一定比例；
7. 实验批改将用随机分配的方式进行；
8. 保留未解释细节的最终解释权；
9. 答辩时未能答对问题会扣掉总体5%~30%的分数。

信号量

背景知识

信号量的定义是理论课的内容，这里不展开讨论，下面是摘自中文维基百科的信号量的相关介绍：

信号量（英语：Semaphore），它以一个整数变量，提供信号，以确保在并行计算环境中，不同进程在访问共享资源时，不会发生冲突。是一种不需要使用忙碌等待的方法。

信号量的概念是由荷兰计算机科学家艾Dijkstra发明的，广泛的应用于不同的操作系统中。在系统中，给予每一个进程一个信号量，代表每个进程目前的状态，未得到控制权的进程会在特定地方被强迫停下来，等待可以继续进行的信号到来。如果信号量是一个任意的整数，通常被称为计数信号量，或一般信号量（general semaphore）；如果信号量只有二进制的0或1，称为二进制信号量（binary semaphore）。在linux系中，二进制信号量（binary semaphore）又称Mutex。

POSIX对信号量的解释

这是对信号量的一些说明，请务必阅读具名信号量相关的段落和原语：

http://linux.die.net/man/7/sem_overview

匿名信号量

POSIX中对匿名信号量的要求是这样的： 线程共享的匿名信号量是存放在线程共享的空间中的，比如全局变量； 进程共享的匿名信号量必须存放在进程的**共享内存区**中。

我们的OSlab实现的是进程，而非线程，因此无法通过全局变量的方式共享信号量； 我们目前也没有实现共享内存，更无法将信号量存放在共享内存区。

因此结论就是我们暂时无法实现匿名信号量。

对此感兴趣的同学可以看看这个问题： http://stackoverflow.com/questions/16400820/c-how-to-use-posix-semaphores-on-forked-processes

这个题主没有好好RTFM，遇到了问题。然后得票最高的答主给题主介绍了 通过共享内存方式使用匿名信号量的方法。

具名信号量

华山一条道，我们现在被迫实现具名信号量了，要求是（从函数名称上）兼容POSIX：

- sem_open：创建一个新的信号量或者打开一个已有的信号量
- sem_close：某一个进程结束对一个信号量的使用之后，便可以用sem_close来关闭该信号量

- sem_wait：相当于P操作，将信号量减1，如果该信号量本来为0，则挂起当前进程
- sem_post：相当于V操作，将信号量加1，如果该信号量本来为0，则唤醒因该信号量而挂起的进程

实现信号量相关的说明和建议

- 我们不要求系统调用的参数兼容POSIX
- 信号量的标识符可以是字符串或数字
- 建议为sem_open增加一个参数用于表示创建二元信号量，方便实现PC问题中的互斥
- 管理因P操作而挂起的进程的可以用专门的链表，也可以用特定的状态描述符
- 遵从kiss法则，你甚至可以为每一个信号量设置一个进程状态， 这样该信号量的V操作唤醒挂起进程时可以单纯地搜索特定的值。
- Linux的PV操作的实现可以作为一个参考（感人的中文资料）：http://www.cnblogs.com/biyeymyhjob/archive/2012/07/21/2602015.html

拓展

下面是可选工作，助教会根据可选工作的完成情况加分。

实现多线程

在 lab 3 中，你实现了fork，用于创建一个与调用者几乎完全相同的进程。 但是前面我们讲到了，纯粹的进程模式让我们无法通过全局变量的方式来使用信号量。 因此为了方便的使用PV操作，我们可以实现一个支持多线程的OS。

为此你需要增加一个系统调用用于创建一个新的线程，该系统调用的参数是一个函数的地址，和传入该函数的参数。

kernel需要做的工作是：

- 为新线程分配一个新的PCB
- 将原有的PCB中的映射拷贝一份（注意这里是浅拷贝）
- 为新的线程分配新的用户栈和内核栈
- 将传入的参数填写到新线程的Trapframe
- 将新线程的Trapframe中的esp置为新的用户栈顶，eip置为函数地址

从工作量来看，线程的创建比进程的创建更加简单，因此鼓励大家实现一下。

多线程下的信号量

这里有一些值得思考的问题，比如：

- 普通的全局变量是不是可以被用户修改?
- 原来的多进程的信号量的通信方式是否可以照搬?
- 系统调用的参数应该进行怎样的调整?
- 如何利用多线程的信号量保证程序按预期执行?

实现共享内存

这部分没有建议的实现方案，有兴趣的同学可以了解两方面的内容：

- Linux的mmap的实现
- JOS的用户处理缺页

如果有同学实现了JOS的COW，可以思考一下， 能不能直接利用或是修改OS提供的映射内存的系统调用来实现用户进程的共享内存?

杂项

不同的框架

由于各位同学使用的代码的框架各不相同，而不同的框架中实现原子操作的方式也各不相同。 我们的讲义假设同学是通过关中断的方法实现的，如果有使用自旋锁的同学需要自行注意 自旋锁的占用和释放的时机。

在jos的代码中是有自旋锁的，但是jos的做法得到的效果事实上和我们的在内核态关中断的做法区别不大， 因为jos的基础实现保证了只有一个进程在内核态运行。

我们鼓励大家将jos的自旋锁移植到自己的OS中， 或者自己实现一个自旋锁，用于进行本实验。不过如果你认为自旋锁的使用较为麻烦，那么你可以 通过关中断的方式实现。

结果展示方式

由于最初大家完成的用户进程都是游戏，而通过游戏展示进程间通信的方式是比较困难的， 因此在本实验的基本要求中，没有游戏相关的部分。 为了展示你实现的进程间通信，你需要通过你的进程间通信方式解决**生产者和消费者问题**。 另外，有不少同学的实验依赖于时间和键盘中断，在lab4中，你需要在特定的部分关中断， 请务必注意。