OSLAB3实验报告

余晨宁 151242062 匡亚明学院
151242062@smail.nju.edu.cn
2017.05.12

实验环境

ubuntu：64位16.04，gcc：5.3.1, 要先apt-get install gcc-4.8-multilib

实验进度

我完成了fork,sleep,exit,getpid等系统调用，但fork仍然存在些许的bug。并用时钟中断驱动进程的调度。有分配进程、初始化trapframe、分配内存、运行进程、释放进程的函数。游戏成功运行。

实验心得和想法

写fork的时候我非常地崩溃。我花了实验中大部分的时间在实现fork调用上，但是fork还是出现了bug。有时会出现page fault，有时则能够成功运行。此外我还很羡慕那些能在网上找到代码然后随便改改就能顺利运行的同学，如果我有那样的搜索水平的话，恐怕我也不会白白浪费这么多时间吧。

即使如此，我还是学到了很多知识，比如深拷贝具体的意思是拷贝内容而并非仅拷贝地址，以及调度是依靠时钟中断完成的。

有没有不依靠时钟中断的调度方法？我觉得在我的这个实现下是不可能的，假设有一个用户程序的内容就是while(1)，那么它除了电脑快没电了或者时钟中断就无法切换到内核态，而调度函数是处于内核里的，因此我认为调度只能够依靠时钟中断。正如上课所说，时钟中断是操作系统处理用户程序的最后一根救命稻草。

我的调度程序很简单：在能够运行的pcb列表按顺序寻找一个非当前进程的pcb，然后运行它。这个调度程序自然是在实际场景中有效率上的缺失，但根据kiss原则，我的用户程序很简单，因此只需要配对这样一个调度程序就足够了。

我将sleep_time作为阻塞时间设置在每个pcb里，每当时钟中断来临一次时，在睡眠的进程的sleep_time就分别减一，当sleep_time为0的时候，该pcb的状态就变为RUNNABLE而非SLEEP。
我看github有的代码的exit是直接把这个进程的状态改为FREE就行了，我觉得这有些浪费资源，因此我先将这些进程的占用的物理页释放掉，再将其状态改为FREE。

我没有实现多线程，因此game也没有多线程化。

总的来说，我通过这次oslab了解到了仅仅看我们的讲义是不够的，还要提高知识水平，去看看人家清华的ucore讲义，再学习一个，才能更加深入地理解进程的管理知识。

助教玩游戏的时候要记得先按enter啊，不然游戏进不去的2333。