oslab3 实验报告

姚荣春 151110079

1. gcc 版本及联系方式

Ubuntu 4.9.3-13ubuntu2

15895873878@163.com

2. 测试方法

在目录 src/test 下是一个测试程序，make run 之后会同时运行 test 和 game 两个程序，test 中 fork 了 5 下然后 sleep 1 秒，打印自己的 pid，最后退出（即测试了所有的功能），相关信息都会输出到标准输出，助教直接看输出窗口即可。所有 test 结束后只剩 game 在运行，如果想单独测试什么其它功能请在 test.c 中更改。usrlib 暂时不能使用，用户程序用的库在当前文件夹中。

3.实验心得

（1）第一个遇到的坑是在写调度算法的时候，用 PCB 记录下了整个的 Trapframe ,但是运行的时候会莫名其妙跳转到内核中，十分奇怪。最后我的同学也用同样的实现方法，出现了这个问题。他发现了问题所在：在用户陷入内核的时候还未等到关中断，这时又来了时间中断，刷新了 Trapframe，从而使得 Trapframe 的 eip 是内核的一个地址，如果这时候恰好时间片用尽，发生程序切换就进入内核，但

是栈的位置不对，嵌套无法恢复，从而崩溃。改进方法：PCB 使用指针记录 Trapframe 位置，或者特判禁止内核态时钟进行进程调度。

（2）第二个遇到的坑是由于 PCB 和 pid 是两个不同资源，最开始想了一个方法，就是进程的 pid 等于 PCB 序号。但是由于我之前的函数实现成进程切换会申请新的 PCB 删除旧的 PCB，这会导致进程更换 PCB 了之后会改变进程的 pid，这显然是不附和要求的。最后给分开实现了，的确是一个败笔，应该实现成根据 pid 使用固定的 PCB，这样就不需要把 PCB 视为一个资源。

（3）最后一个坑在于维护 PCB 链表的时候，删除等待序列中的进程把它加入可执行队列时忘记考虑进程是链表第一个进程的情况。

（4）第一个坑是很难想象的。体会：有时候知道 bug 在哪但是不会调就重写绕过 bug 吧。

4. 思考题

（1）你应该注意到了，我们的数据结构中没有存放页框的物理地址，那如何根据一个 page_info 元素 找到它对应的页框呢？

答：第 i 个 page_info 代表(i-1)*4096 起始的物理地址。

（2）能不能根据物理页的起始地址找到它对应的数据结构呢？

答：能，第(paddr/4096)个数据结构（从 1 开始算）

5.makefile 使用介绍

make run 编译运行

make gdb 编译且打开 gdb

make clean 清除.o 等文件

make run 之后运行 terminal.sh(需要加权限)可以连接 qemu