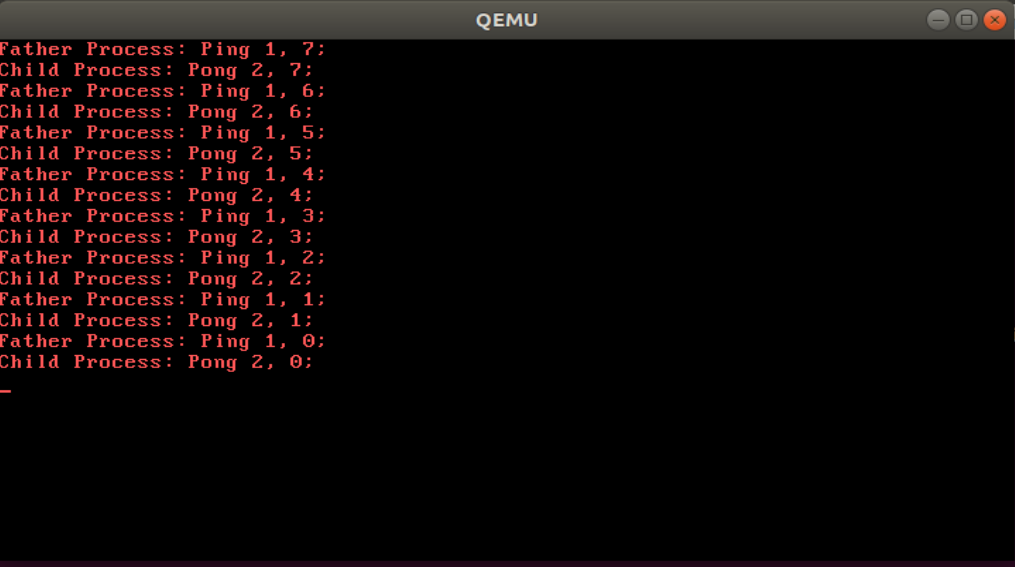
姓名：张涵之 学号：191220154 邮箱：[1683762615@qq.com](mailto:1683762615@qq.com)

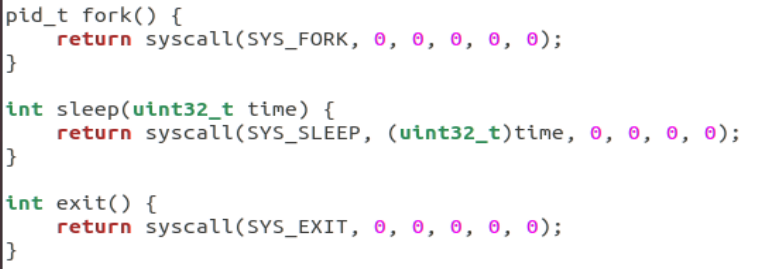
实验进度：我完成了3.1-3.3的必做内容，没有涉及选做内容

实验结果：下图为官方测试代码运行效果



实验修改的代码位置：

3.1. 完成库函数 lab3/lib/syscall.c

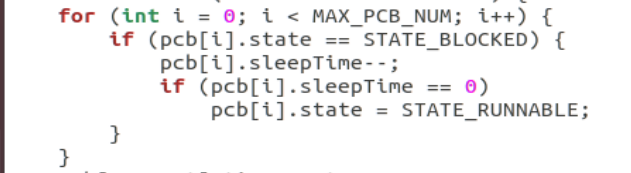


仿照实验2调用syscall完善库函数

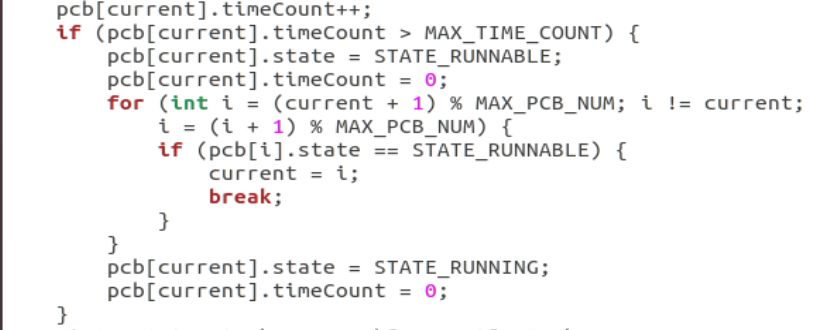
3.2. 时钟中断处理 lab3/kernel/kernel/irqHandle.c

定义时钟中断处理函数timerHandle，其功能为：

1. 遍历pcb中的进程，将状态为STATE\_BLOCKED的进程的sleepTime减一，如果某个进程的sleepTime变为0，其状态重新设为STATE\_RUNNABLE



1. 将当前进程的timeCount加一，如果时间片用完（timeCount==MAX\_TIME\_COUNT）且有其它状态为STATE\_RUNNABLE的进程，切换，否则继续执行当前进程



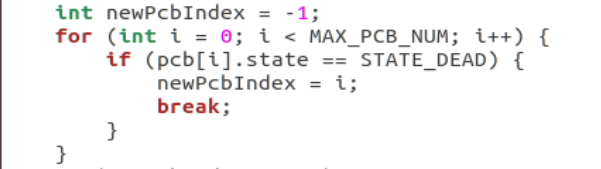
使用手册中提供的参考代码进行进程切换

理解irqHandle对比lab2增加了部分保存与恢复的内容，其具体功能为在执行系统调用前后分别保存和恢复当前进程的esp栈指针位置，并实现内存的分配与回收

3.3. 系统调用例程

1. syscallFork的实现 lab3/kernel/kernel/irqHandle.c

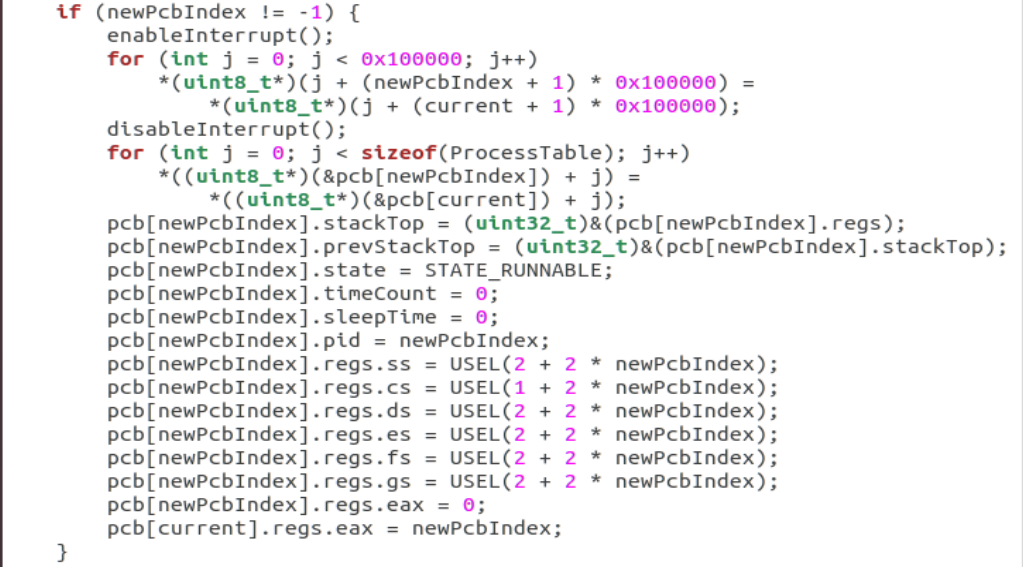
寻找一个空闲的pcb做为子进程的进程控制块



将父进程的资源复制给子进程，代码段和数据段完全拷贝，且需在中断状态下进行

复制pcb时，先将父进程的内容直接复制，再仿照initProc中对pcb[1]的初始化进行修改：其中，栈顶指针需根据父进程（已复制）进行设置，运行状态、timeCount、sleepTime和pid与父进程无关，ss、cs等段寄存器的数值需手动计算

Fork成功，子进程返回值设为0，父进程返回值设为子进程的pid

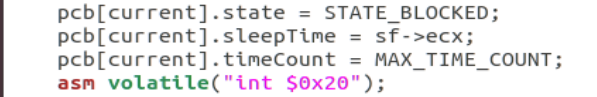


如果没有空闲pcb，则fork失败，父进程返回值设为-1



1. syscallSleep的实现 lab3/kernel/kernel/irqHandle.c

将当前的进程的sleepTime设置为传入的参数，状态设置为STATE\_BLOCKED，然后利用asm volatile("int $0x20"); 模拟时钟中断，利用timerHandle进行进程切换



对传入的参数要进行合法性判断，在这里体现为时间必须是正数

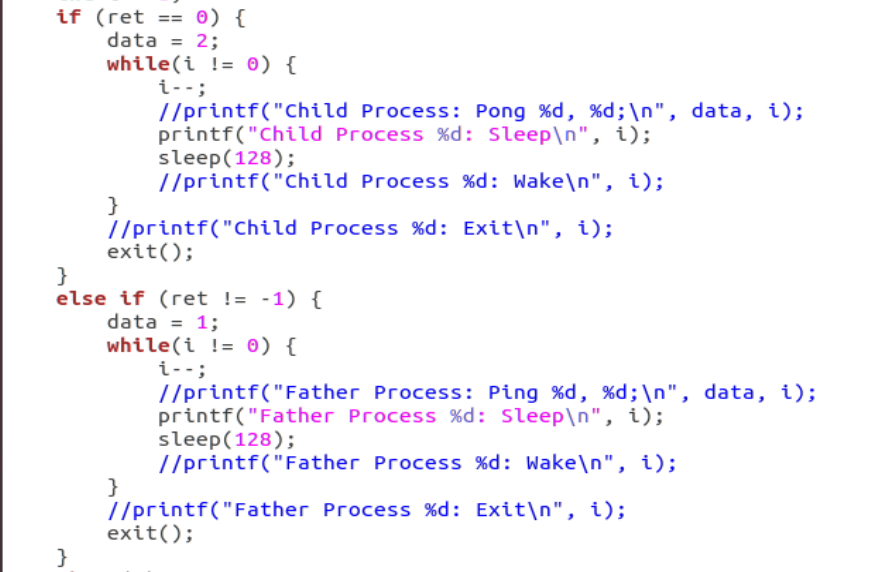
1. syscallExit的实现 lab3/kernel/kernel/irqHandle.c

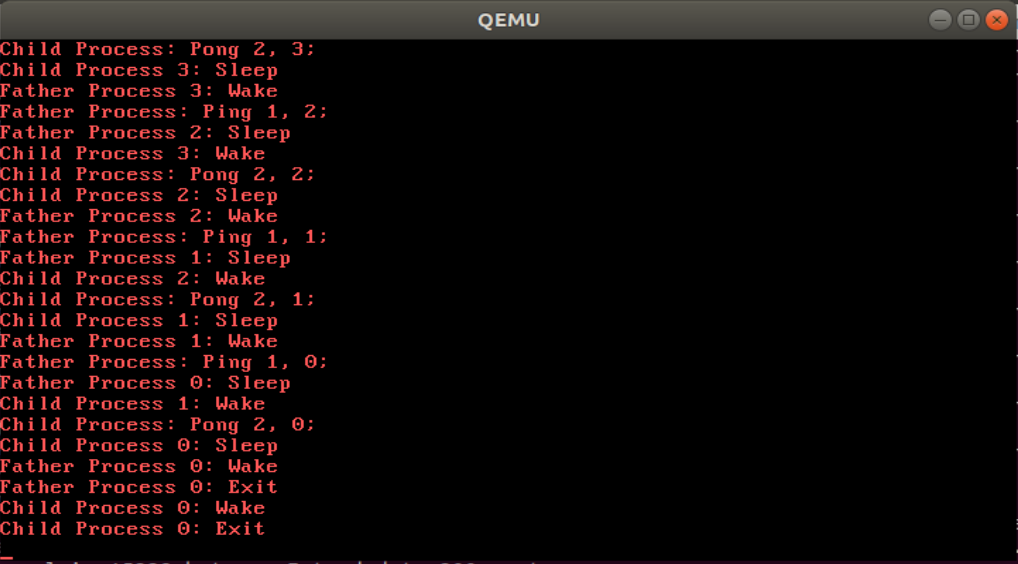
将当前进程的状态设置为STATE\_DEAD，然后模拟时钟中断进行进程切换



思考和总结：

因为框架代码和汇编指令比较陌生，而且文件夹较多，几乎每次开始写实验的时候都觉得无从下手。大概没有别的捷径，只能耐心阅读和理解框架代码，结合手册进行理解。





利用调试输出语句，结合用户程序代码可以验证调用顺序符合逻辑。