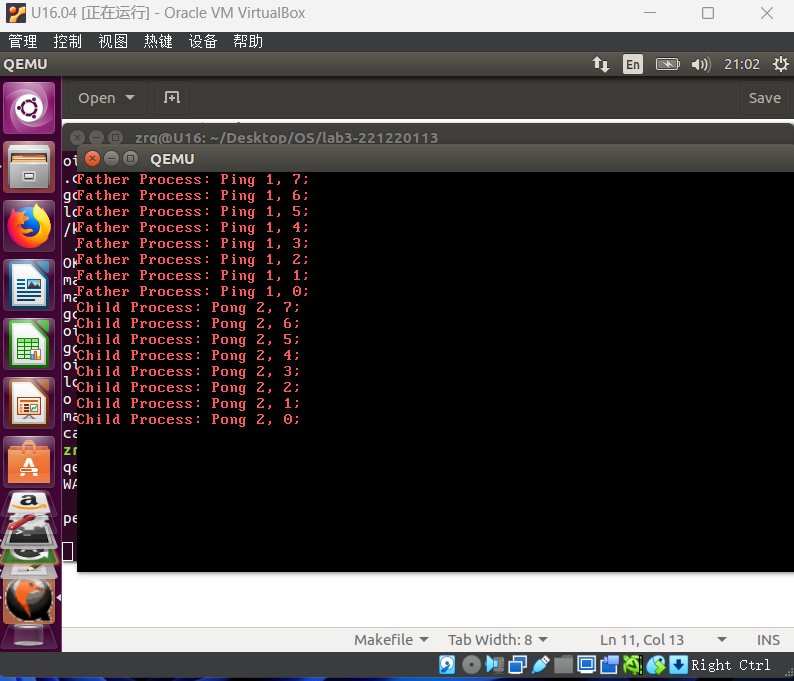
Lab3实验报告

221220113

首先上实验结果



接下来对实验中做的修改进行解释

3.1完善syscall

和lab2完全一样。

首先在用户接口处填好调用号，然后在syscallHandle处调用对应的处理例程即可。

为fork、sleep、exit分别分配1，2，3号。

函数的实现在下面说明。

3.2时钟中断处理timerHandle

在这里，对IDLE做特殊对待：

除非没有其他进程可选，否则不会选择IDLE；若当前进程是IDLE，则无视sleepTime，总是尝试进行轮转。

根据实验报告，整理出：

pcb[0]是内核IDLE，1~MAX\_PCB\_NUM是用户进程。所以特殊对待pcb[0]即可。

timerHandle的基本流程如下:  
为所有STATE\_BLOCK进程减少sleep时间，并进行相应的向STATE\_RUNNABLE转换的处理；

累计当前进程时间片，若需要更换进程或者当前进程是IDLE：

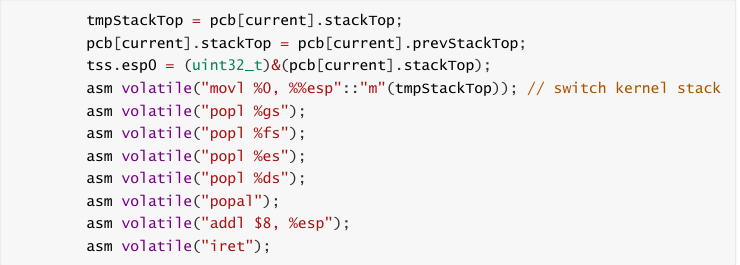
顺序轮转寻找下一个STATE\_RUNNABLE进程next（除非没有其他STATE\_RUNNABLE进程，否则不会选择IDLE）

将current进程转变为STATE\_RUNNABLE并将时间片归零

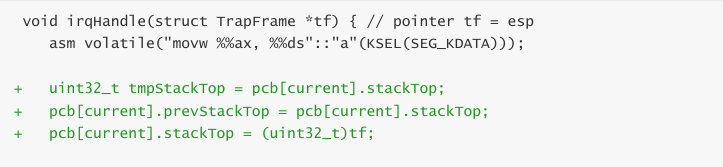
将current设为next

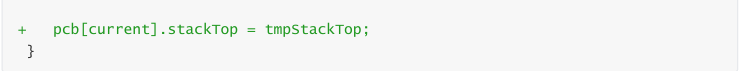
激活current进程，即转变为STATE\_RUNNING

另外，转换进程代码的含义解释如下：



由于第二行，这段代码的指向比较难懂，应该结合另一处prevStackTop出现的代码进行理解：





可以看到，在进行中断处理的时候，将中断的stackFrame作为新的stackTop，而旧的stackTop存储于prevStackTop。

那么prevStackTop的作用很显然就是用于中断时，存储用户堆栈信息：因为stackTop被占用去存储中断现场（以备中断嵌套）了。

那么可以分两种情况：

1. 要启动的进程未进行系统调用，那么此时stackTop==prevStackTop，转换进程代码的前两行就可以忽略了，就是在进行：改写tss，弹出用户栈信息，跳转执行用户程序。
2. 要启动的进程之前被中断嵌套了，那么就激活中断现场，并将用户堆栈信息弹回stackTop

3.3.1syscallFork

首先要找一个STATE\_DEAD的pcb分配给新进程

然后将父进程的一切(length = 0x100000)复制给子进程

将父进程pcb的一切(length = sizeof(processTable))也复制给子进程pcb

下面是要区别的地方（都可以去init\_proc找到）：

//开始调整pcb

            pcb[i].stackTop = (uint32\_t)&(pcb[i].regs);

            pcb[i].prevStackTop = (uint32\_t)&(pcb[i].stackTop);

            pcb[i].state = STATE\_RUNNABLE;

            pcb[i].timeCount = 0;

            pcb[i].sleepTime = 0;

            pcb[i].pid = i;

            //esp不变

            //eflags不变

            //eip不变

            pcb[i].regs.cs = USEL(2 \* i + 1);

            pcb[i].regs.ss = USEL(2 \* i + 2);

            pcb[i].regs.ds = USEL(2 \* i + 2);

            pcb[i].regs.es = USEL(2 \* i + 2);

            pcb[i].regs.fs = USEL(2 \* i + 2);

            pcb[i].regs.gs = USEL(2 \* i + 2);

            pcb[i].regs.eax = 0;    //子进程返回0

            pcb[current].regs.eax = i;  //主进程返回i

3.3.2syscallSleep

找到参数：ecx，根据ecx设置sleep时间，转变为STATE\_BLOCKED后，复用timerHandle即可。

3.3.3syscallExit

转变为STATE\_DEAD后，复用timerHandle即可。