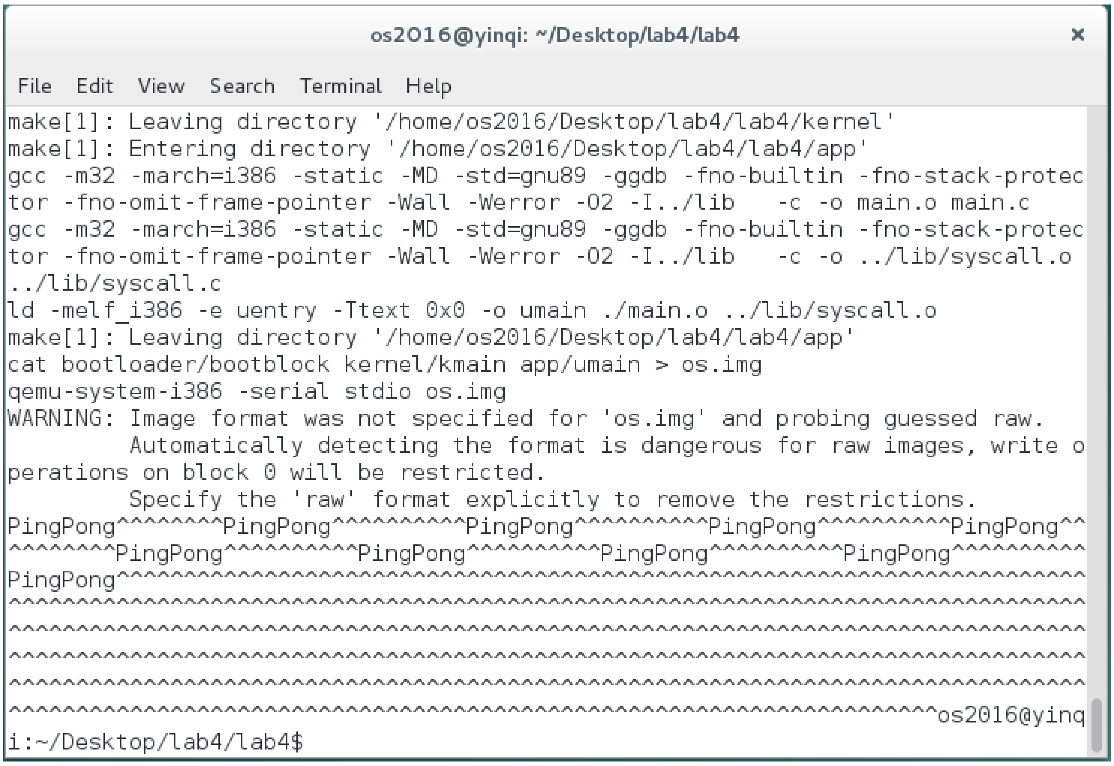
操作系统 Lab4 实验报告

141220132@smail.nju.edu.cn　141220132　银琦

实验进度

完成了实验，使用信号量和 PV 操作完成了 ping、pong 的输出，添加了系统调用 createSem, lockSem, unlockSem, destroySem。

实验结果截图：



实验过程及关键代码截图：

1. 加载用户程序
2. 时钟中断切换到用户进程

3. app/main.c 中的代码如下

```
        createsem();
        if (fork() == 0) {
                int i;
                for (i = 0; i < 10; i++) {
                        locksem();
                        printf("Pong");
//                      sleep(1);
                }

        }
        else {
                int i;
                for (i = 0; i < 10; i++) {
                    printf("Ping");
                        unlocksem();
                        sleep(1);
                }
                destroysem();
        }

    exit(0);
//    while(1);/* {printf("p");sleep(1);}*/
```

4. 用户进程调用 createSem 创建信号量。

   createSem、destroySem、P、V 操作的函数如下：

```
        void P() {
        //      putchar('e');
                s.value--;
                if (s.value < 0)
                        W();
        }
        void V() {
        //      putchar('a');
                s.value++;
                if (s.value <= 0)
                        R();
        }
        void createsem() {
                s.value = 0;
                s.state = Enable;
        }
        void destroysem() {
                s.state = Disable;
        }
```

5. 用户进程通过 fork 产生父子进程。

6. 父进程打印 ping 后调用 unlockSem，并 sleep

unlockSem 对应 V 操作，V 操作中，当 value<=0 时，需要唤醒一个进程，唤醒进程先将被阻塞的进程从 list 队列中取出，然后与 schedule 里当 time_count 减到 0 的时候相似，代码如下：

```
void R() {
//      putchar('b');
        PCB *p = s.list;
        if (s.list->next == NULL) {
//      putchar('f');
                list_del(p);
                s.list = NULL;
        }
        else {
//      putchar('g');
                s.list = s.list->next;
                list_del(p);
        }

        p->state = Runnable;
        PCB *q = Ready;
        if (Ready == NULL)
                Ready = p;
        else {
                while(q->next != NULL)
                        q = q->next;
                list_add_after(q, p);
        }
        current = Ready;
        current->state = Running;
        current->time_count = 10;

}
```

7. 子进程调用 lockSem 后再打印 pang，子进程不 sleep

lockSem 对应 P 操作，P 操作中，当 value<0 时，需要阻塞当前进程，将其从 Ready 队列中删除，放入 list 队列，然后再将 current 置为 Ready，代码与 sleep 函数的代码相似，代码如下：

```
void W() {
        current->sleep_time = 0xffffffff;
        current->state = Blocked;
        PCB* p = s.list;

        if (current->next == NULL) {
                list_del(current);
                Ready = NULL;
        }
        else if (current->next != NULL) {
                Ready = current->next;
                list_del(current);
                current->next = NULL;
        }

        if (s.list == NULL) {
//              putchar('c');
                s.list = current;
                s.list->next = NULL;
        }
        else {
//              putchar('d');
                while(p->next != NULL)
                        p = p->next;
                list_add_after(p,current);
        }

        if (Ready == NULL) {
                current = &idle;
        }
        else {
                current = Ready;
                current->time_count = 10;
                current->state = Running;
                tss.esp0 = (int)current->stack + 4096;
        }
}
```

8. 最后父进程调用 destroySem 销毁进程

对进程设置了两个状态，Enable 和 Disable，Enable 表示打开信号量，Disable 表示关闭信号量，所以最后将进程的状态置为 Disable 即可。

9. 进入 IDLE 线程，循环打印'^'。

遇到的问题：

基本功不太好，所以在定义结构的时候出现了小问题，浪费了一些时间，在完成 lab4 的时候，注意到了 lab3 中出现的问题，所以很快就写完了。

收获心得：

1. 基本功最重要 •••••• 不然很浪费时间 T^T。
2. Lab 基本写完了，之前 PA 没能够完成，总觉得自己没有抱到大腿，在写 lab 的时候，lab2 问了别人思路是什么，在别人的帮助下理解了要做什么，然后自己写代码，调 bug，逻辑方面的错误最后都是自己解决的，lab3 的开头是大神带的，后面都是自己尝试着写的，一步步输出，一步步调试，觉得长进了不少，最后觉得如果可以的话，多花些时间看讲义看书查资料写代码比抱大腿直接知道怎么写靠谱的多！可能正是因为之前认认真真调 bug，lab4 在自己独立完成的情况下并没有花费太多时间，实验比较简单是一方面，另一方面也需要对实验内容和代码结构多多理解。