Lab4 实验报告

141242026
刘驭壬
邮箱：141242026@smail.nju.edu.cn

实验进度：

我完成了Lab4的所有内容。其中包括

1）我实现了关于信号量操作的系统调用：sem_open,sem_close,sem_wait,sem_post.

2）我实现了多线程（pthread_create，pthread_free函数）。

关于实验的说明

1.在sem_wait和sem_post中我维护了一个挂起进程的链表。

2.测试函数是pc.c中的pc_main函数（一个生产者消费者问题）。

3.我的线程的栈是开在进程栈上，从进程的栈中选一部分没有用过的作为线程栈

```
int pthread_create(void *p){
        //int i;
        envid_t p_id=curenv->env_id;
        struct Env *env=NULL;
        int judge=env_alloc(&env,p_id);
        if(judge!=0){
                printk("env_alloc error!");
                return judge;
        }
        env->env_pgdir=curenv->env_pgdir;
        curenv->threadnum++;
        //env->env_tf=curenv->env_tf;
        env->env_tf.esp=USTACKTOP - curenv->threadnum * 2 * PGSIZE + PG
SIZE + 0x60;
        //printk("function eip=%x\n",(uint32_t)p);
        env->env_tf.eip=(uint32_t)p;
        //printk("tf eip=%x\n",env->env_tf.eip);
        env->env_tf.eax=0;//as the return value of process
        return env->env_id;
}
```

我遇到的问题：

1.我对sem_wait函数是这样实现的：如果此时信号量已经为0，则不减一，直接挂起。（即认为信号量不能为负），一开始实现的配套的sem_post函数一定会加一，这样的实现是有问题，因为挂起的时候没有减一，而post的时候一定会加一。之后改为当post的时候先检测wait_list链表是否为空，如果为空则post的时候加一，否则post的时候不加一，并直接唤醒一个挂起进程。

sem_wait中相关操作：

```
        if(semaphore[index].binary==false){
                if(!semaphore[index].wait_list){
                        semaphore[index].num++;
                        return ;
                }
                else{
                        curenv->env_status=ENV_RUNNABLE;
                        struct Env*a=semaphore[index].wait_list;
                        semaphore[index].wait_list=semaphore[index].wai
t_list->env_link;
                        a->env_status=ENV_RUNNABLE;
                        env_run(a);
                        return ;
                }
        }
```

sem_post中相关操作：

```
      if(semaphore[index].binary==false){
              int temp=semaphore[index].num;
              //printk("semaphore num=%d\n",temp);
              if(temp==0){
                      curenv->env_status=ENV_NOT_RUNNABLE;
                      curenv->env_link=semaphore[index].wait_list;
                      semaphore[index].wait_list=curenv;
                      struct Env* a=seek_next_runnable();
                      env_run(a);
                      return 0;
              }
              else{
                      semaphore[index].num--;
                      return temp;
              }
```

2.由于线程的栈是直接开在进程栈上的，进程栈区要开得比较大，一开始进程栈只开了一页，所以一直缺页，后来直接改成了4M。