OSLab4 实验报告

姓名：凌嘉伟 学号：151220061
邮箱：151220061@smail.nju.edu.cn

一、实验进度

Debug OSLab3 基本完成
实现了线程的操作，编写了 thread_create()和 thread_free()两个函数
实现了信号量的操作，基本实现了 sem_init(), sem_destroy(), sem_wait(), sem_post()函数
编写了生产者和消费者问题的程序并进行测试

二、实验讲义问题

1、普通的全局变量是不是可以被用户修改？

普通的全局变量具有全局作用域(文件作用域)，通常在整个工程中有效，定义在一个文件中，在其他文件中使用 extern 声明即可使用。因此用 extern 声明即可修改。

2、 linux 进程和线程下匿名信号量的实现有什么本质区别？

线程 :如果信号量是在多个线程中共享，那么它所在的内存区应该是这些线程应该都能访问到的全局变量或者 malloc 分配到的内存。

进程 :如果是在多个进程间共享, 那么这段内存应该本身是一段共享内存(使用 mmap、shmget 或 shm_open 申请的内存)

三、实验过程及注意说明

1、 关于 sem_init 函数（该函数存放于 kernel/semaphore/semaphore.c 中）

将信号量初始化，将信号量有关的变量赋值为 0（或 false）

2、 关于 sem_wait 和 sem_post 函数（该函数存放于 kernel/semaphore/semaphore_func.c 中）：

根据 sem_wait 和 sem_post 的操作定义：

sem_wait：相当于 P 操作，将信号量减 1，如果该信号量本来为 0，则挂起当前进程/线程

sem_post：相当于 V 操作，将信号量加 1，如果该信号量本来为 0，则唤醒因该信号量而挂起的进程/线程

因此在 sem_wait 和 sem_post 函数中，实现了一个用来挂起进程的链表。

在 wait 操作中，先判断该信号量是否为 0，若为 0，则直接挂起（否则减 1）：

```
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;
}
```

在 post 操作中，我们先检测 wait_list 是否为空（防止挂起的时候因为没有减 1 但 post 必定加 1 这种情况），然后若信号量为 0 则唤醒一个被挂起的线程：

```
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].wait_list->env_link;
    a->env_status = ENV_RUNNABLE;
    env_run(a);
    return;
}
```

此外，我们在 sem 函数操作中还判断是否为二进制信号量，用来同步或互斥。

3、 关于 sem_destroy 和 sem_trywait

实现了 destroy 之后发现没有用到…trywait 因为没有用到就没有实现了…

4、 关于生产者和消费者问题（存放在 game/test_semaphore.c 中）

在 makefile 中，通过修改 game 入口来检测生产者和消费者问题：

```
$(GAME): $(GAME_O) $(LIB_O)
    $(LD) -m elf_i386 -e test_main --nostdlib -o $@ $^ $(shell $(CC) $(CFLAGS) --print-libgcc-file-
name)
```

5、 关于 lab3 的 debug

在 lab3 中实现的 fork、sleep、exit 函数，由于当时仍然有问题，因此并没有在游戏的时候加入这些函数，现在补充测试，通过修改 makefile 中的 game 入口比如 fork_test、sleep_test、exit_test 来测试_(:3 」∠)_

四、实验感想

又是在急急忙忙中写的，有些时候完全没有思路因此很纠结……也 debug 了不少时间，实验报告也不知道怎么写……总之 lab 完成比较乱，实验报告也完成的很乱，心塞啊……TUT