

## Lab4 进程同步

### 1、在内核中定义 semaphore 结构体

在 kernel/include 中新建 semaphore.h 文件，文件内容为：

```
1  #include "x86.h"
2  #include "sched.h"
3  #define MAX_SEM 10
4  typedef struct Semaphore
5  {
6      int value;
7      int id;
8      int ifuse;
9      PCB *next;
10 }Semaphore;
11
12 Semaphore semaphore[MAX_SEM];
13 void P(Semaphore *s);
14 void V(Semaphore *s);
15 void seminit();
16 int getsem();
```

Value 为信号量的值，id 为信号量 id，ifuse 表示信号量是否正在使用，next 为当前阻塞在该信号量上的进程链表。

定义了大小为 10 的信号量数组，等待用户使用。

定义操作 P，V。

定义函数 seminit，作用为初始化所有信号量。

定义函数 getsem，作用为获取一个可用信号量 id

### 2、具体函数实现

PV 操作均在课程指导上。

P 操作中调用的 W：

```

4 void W(Semaphore *s)
5 {
6     if(s->next==NULL)
7     {
8         s->next=current;
9         current->next=NULL;
10        current->prev=NULL;
11        current->state = BLOCKED;
12        current = delfirst(&runnable);
13        if(current == NULL)
14            current = &idle;
15    }
16    else
17    {
18        PCB *temp=s->next;
19        while(temp->next!=NULL)
20        {
21            temp=temp->next;
22        }
23        temp->next=current;
24        current->state=BLOCKED;
25        current->prev=temp;
26        current->next=NULL;
27        current=delfirst(&runnable);
28    }
29 }

```

作用为将当前进程阻塞在信号量 s 上

V 操作调用的 R 函数：

```

31 void R(Semaphore *s)
32 {
33     if(s->next==NULL)
34         return;
35     else
36     {
37         PCB *temp=s->next;
38         s->next=s->next->next;
39         if(s!=NULL)
40             s->next->prev=NULL;
41         temp->state=RUNNABLE;
42         temp->next=NULL;
43         temp->prev=NULL;
44         addpcb(&runnable,temp);
45     }
46 }

```

作用为释放 s 信号量上阻塞的一个进程。

### 3、实现系统调用

在 lib/types.h 中定义结构 sem\_t:

```

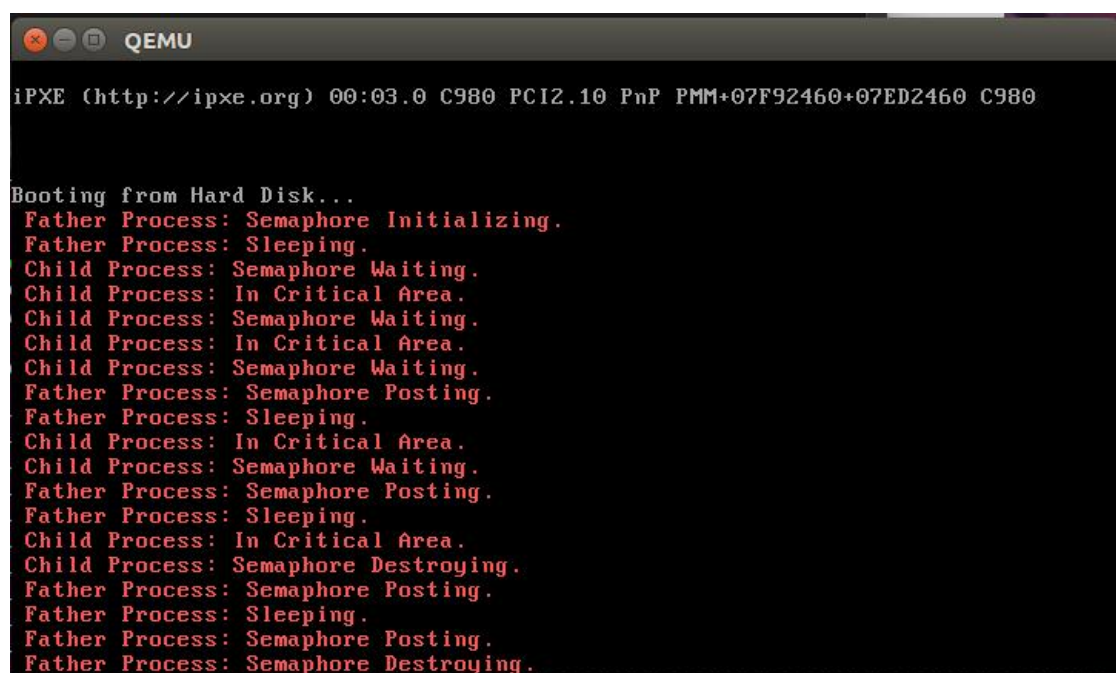
struct sem_t
{
    int value;
    int id;
};
typedef struct sem_t sem_t;

```

分别调用 syscall 实现 sem\_init,sem\_wait,sem\_post,sem\_destroy 函数并在 lib.h 中声明。

在内核的 irqHandle 函数中实现上述系统调用。分别调用相关函数实现。

#### 4、实验结果：



```

QEMU
iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F92460+07ED2460 C980

Booting from Hard Disk...
Father Process: Semaphore Initializing.
Father Process: Sleeping.
Child Process: Semaphore Waiting.
Child Process: In Critical Area.
Child Process: Semaphore Waiting.
Child Process: In Critical Area.
Child Process: Semaphore Waiting.
Father Process: Semaphore Posting.
Father Process: Sleeping.
Child Process: In Critical Area.
Child Process: Semaphore Waiting.
Father Process: Semaphore Posting.
Father Process: Sleeping.
Child Process: In Critical Area.
Child Process: Semaphore Destroying.
Father Process: Semaphore Posting.
Father Process: Sleeping.
Father Process: Semaphore Posting.
Father Process: Semaphore Destroying.

```

Ubuntu 版本: 16.04

Gcc 版本: 5.4.0 20160609

Qemu 版本:

QEMU emulator version 2.5.0 (Debian 1:2.5+dfsg-5ubuntu10.14)