

实验内容

共享内存，使得多个进程可以访问同一块内存空间，是最快的可用IPC形式。是针对其他通信机制运行效率较低而设计的。往往与其它通信机制，如信号量结合使用，来达到进程间的同步及互斥。在本次实验中，使用共享内存实现进程间的通讯，并使用信号量来实现对共享内存数据的保护。

实验内容

实验文件

1	common.c	// 共享函数
2	common.h	// 共享头文件
3	money_without_sem.c	// 没有使用信号量的通讯
4	money_without_sem.out	// linux 可执行文件
5	money_with_sem.c	// 使用信号量的通讯
6	money_with_sem.out	// linux 可执行文件

编译和执行的指令

1. 未使用信号量

编译: `gcc common.h common.c money_without_sem.c -o money_without_sem.out`

执行: `./money_without_sem.out`

2. 使用信号量

编译: `gcc common.h common.c money_with_sem.c -l pthread -o money_with_sem.out`

执行: `./money_with_sem.out`

1. 创建通用函数

在 `common.c` 中创建

```
1  int create_ipc(int); // 创建共享内存
2  int get_ipc(int); // 获得共享内存的id
3  int destory_ipc(int); // 销毁共享内存
4
5  int get_sem(int num); // 获得 num 个信号量
6  int set_sem_val(int id, int index, int val); // 给第 index 个信号量赋初值
7  int del_sem(int id, int index); // 删除信号量
8  int semwait(int id, int index); // 第 index 个信号量 P 操作
9  int sempost(int id, int index); // 第 index 个信号量 V 操作
```

2. 未使用信号量

在 `fork()` 后, 对同一个信号量进行 2000 次 +1 操作

```
1  while (count++ < 2000)
2  {
3      balance = (int *)shmat(mid, NULL, 0);
4      tmp = *balance;
```

```

5
6     printf("%15s-%4d-get:%d\n", this, count, *balance);
7     *balance = tmp + 1;
8     printf("%15s-%4d-put:%d\n", this, count, *balance);
9     if (shmdt(balance) == -1)
10    {
11        // 取消映射
12        printf("shmdt() error\n");
13    }
14 }

```

3. 使用信号量

在未使用信号量的基础上，添加信号量，对临界资源 `balance` 进行保护

```

1  while (count++ < 2000)
2  {
3      semwait(sems_id, 1); // P 操作
4      balance = (int *)shmat(mid, NULL, 0);
5      tmp = *balance;
6      printf("%15s-%4d-get:%d\n", this, count, *balance);
7
8      *balance = tmp + 1;
9      printf("%15s-%4d-put:%d\n", this, count, *balance);
10     if (shmdt(balance) == -1)
11     {
12         // 取消映射
13         printf("shmdt() error\n");
14     }
15     sempost(sems_id, 1); // V 操作
16 }

```

实验结果

1. 未使用信号量，最终结果并不是 4000

```

Child process-1994-put:2240
Child process-1995-get:2240
Child process-1995-put:2241
Child process-1996-get:2241
Child process-1996-put:2242
Child process-1997-get:2242
Child process-1997-put:2243
Child process-1998-get:2243
Child process-1998-put:2244
Child process-1999-get:2244
Child process-1999-put:2245
Child process-2000-get:2245
Child process-2000-put:2246
释放共享内存-2883590

```

2. 使用信号量，保证了最终结果为4000

```
Parent process-1996-get:3991
Parent process-1996-put:3992
Child process-1997-get:3992
Child process-1997-put:3993
Parent process-1997-get:3993
Parent process-1997-put:3994
Child process-1998-get:3994
Child process-1998-put:3995
Parent process-1998-get:3995
Parent process-1998-put:3996
Child process-1999-get:3996
Child process-1999-put:3997
Parent process-1999-get:3997
Parent process-1999-put:3998
Child process-2000-get:3998
Child process-2000-put:3999
Parent process-2000-get:3999
Parent process-2000-put:4000
释放共享内存-2850822
释放信号量集-393217
```