经典IPC问题实现作业报告

软工2班 3017218092 侯雨茜

一、运行实例代码

进入文件目录，编译运行

```
cd /home/yuqianhou/文档/pc_semaphore
make
./init
```

先运行生产者：

```
./producer
```

再新建一个shell，运行消费者：

```
./consumer
```

运行结果如下：

```
yuqianhou@yuqianhou-VirtualBox:~/文档/pc_semaphore$ ./init
yuqianhou@yuqianhou-VirtualBox:~/文档/pc_semaphore$ ./producer
Producing item 1
Inserting item 1
Producing item 2
Inserting item 2
Producing item 3
Inserting item 3
Producing item 4
Inserting item 4
Producing item 5
Inserting item 5
Producing item 6
Inserting item 6
Producing item 7
Inserting item 7
Producing item 8
Inserting item 8
Producing item 9
Inserting item 9
Producing item 10
Inserting item 10
Producing item 11
Inserting item 11
Producing item 12
Inserting item 12
Producing item 13
Inserting item 13
Producing item 14
Inserting item 14
Producing item 15
Inserting item 15
Producing item 16
Inserting item 16
Producing item 17
Inserting item 17
Producing item 18
Inserting item 18
Producing item 19
```

```
Inserting item 19
Producing item 20
Inserting item 20
Producing item 21
Inserting item 21
Producing item 22
Inserting item 22
Producing item 23
Inserting item 23
Producing item 24
Inserting item 24
Producing item 25
Inserting item 25
Producing item 26
Inserting item 26
Producing item 27
Inserting item 27
Producing item 28
Inserting item 28
Producing item 29
Inserting item 29
Producing item 30
Inserting item 30
Finish!
```

```
yuqianhou@yuqianhou-VirtualBox:~$ cd /home/yuqianhou/文档/pc_semaphore
yuqianhou@yuqianhou-VirtualBox:~/文档/pc_semaphore$ ./consumer
Removing item 1
Consuming item 1
Removing item 2
Consuming item 2
Removing item 3
Consuming item 3
Removing item 4
Consuming item 4
Removing item 5
Consuming item 5
Removing item 6
Consuming item 6
Removing item 7
Consuming item 7
Removing item 8
Consuming item 8
Removing item 9
Consuming item 9
Removing item 10
Consuming item 10
Removing item 11
Consuming item 11
Removing item 12
Consuming item 12
Removing item 13
Consuming item 13
Removing item 14
Consuming item 14
Removing item 15
Consuming item 15
Removing item 16
Consuming item 16
Removing item 17
Consuming item 17
Removing item 18
Consuming item 18
Removing item 19
Consuming item 19
Removing item 20
Consuming item 20
```

二、读者优先描述

在几个经典IPC问题中，我选择了读者写者问题中的读者优先进行编程。

读者优先描述如下：

如果读者来：

1. 无读者、写者，新读者可以读；
2. 无写者等待，但有其他读者正在读，新读者可以读；
3. 有写者等待，但有其他读者正在读，新读者可以读；

4. 有写者写，新读者等

如果写者来：

1. 无读者，新写者可以写；
2. 有读者，新写者等待；
3. 有其他写者写或等待，新写者等待

三、增加读者和写者

新建 `reader.c` 文件，编写如下代码：

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <sys/shm.h>
#include <sys/msg.h>
#include "myipc.h"

int main(int argc, char *argv[])
{
    int i,item,shmid;
  // semaphore mutex,empty,full;
    semaphore fmutex = 1;
    semaphore rdcntmutex = 1;
    int reader_count = 0;
    union semun sem_union;
  void *shared_memory = (void *)0;
  struct shared_use_st *shared_stuff;

  if ( (fmutex = semget((key_t)KEY_MUTEX,1,0666|IPC_CREAT)) == -1 ) {
    fprintf(stderr,"Failed to create semaphore!");
    exit(EXIT_FAILURE);
  }
  if ( (rdcntmutex = semget((key_t)KEY_EMPTY,1,0666|IPC_CREAT)) == -1
) {
    fprintf(stderr,"Failed to create semaphore!");
    exit(EXIT_FAILURE);
  }
  // if ( (full = semget((key_t)KEY_FULL,1,0666|IPC_CREAT)) == -1 ) {
  //  fprintf(stderr,"Failed to create semaphore!");
  //  exit(EXIT_FAILURE);
  // }
```

```
  if ( (shmid = shmget((key_t)KEY_SHM,sizeof(struct shared_use_st),0666|IPC_CREAT)) == -1 ) {
    fprintf(stderr,"Failed to create shared memory!");
    exit(EXIT_FAILURE);
  }

  if ( (shared_memory = shmat(shmid,(void *)0,0) ) == (void *)-1) {
    fprintf(stderr,"shmat failed\n");
    exit(EXIT_FAILURE);
  }
  shared_stuff = (struct shared_use_st *)shared_memory;

    for (i = 0; i < 30; i++)
    {
        item = ++(shared_stuff->cur);
    sleep(1);
    printf("Reading %d\n",item);

        sem_p(rdcntmutex);
        if (reader_count == 0)
        {
            sem_p(fmutex);
        }
        reader_count += 1;
        sem_v(rdcntmutex);

        // Do read operation
        (shared_stuff->buffer)[(shared_stuff->hi)]=item;
    (shared_stuff->hi) = ((shared_stuff->hi)+1) % BUFFER_SIZE;
    printf("Inserting item %d\n",item);

        sem_p(rdcntmutex);
        reader_count -= 1;
        if (reader_count == 0)
        {
            sem_v(fmutex);
        }
        sem_v(rdcntmutex);

    }

    if (shmdt(shared_memory) == -1) {
          fprintf(stderr, "shmdt failed\n");
    exit(EXIT_FAILURE);
  }
  printf("Finish!\n");
  getchar();
    exit(EXIT_SUCCESS);
}
```

新建 `writer.c` 文件，编写如下代码：

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <sys/shm.h>
#include <sys/msg.h>
#include "myipc.h"

int main(int argc, char *argv[])
{
    int i,item,shmid;
  // semaphore mutex,empty,full;
    semaphore fmutex = 1;
    semaphore rdcntmutex = 1;
    int reader_count = 0;
    union semun sem_union;
  void *shared_memory = (void *)0;
  struct shared_use_st *shared_stuff;

  if ( (fmutex = semget((key_t)KEY_MUTEX,1,0666|IPC_CREAT)) == -1 ) {
    fprintf(stderr,"Failed to create semaphore!");
    exit(EXIT_FAILURE);
  }
  if ( (rdcntmutex = semget((key_t)KEY_EMPTY,1,0666|IPC_CREAT)) == -1
) {
    fprintf(stderr,"Failed to create semaphore!");
    exit(EXIT_FAILURE);
  }
  // if ( (full = semget((key_t)KEY_FULL,1,0666|IPC_CREAT)) == -1 ) {
  //  fprintf(stderr,"Failed to create semaphore!");
  //  exit(EXIT_FAILURE);
  // }
  if ( (shmid = shmget((key_t)KEY_SHM,sizeof(struct
shared_use_st),0666|IPC_CREAT)) == -1 ) {
    fprintf(stderr,"Failed to create shared memory!");
    exit(EXIT_FAILURE);
  }

  if ( (shared_memory = shmat(shmid,(void *)0,0) ) == (void *)-1) {
    fprintf(stderr,"shmat failed\n");
    exit(EXIT_FAILURE);
```

```
  }
  shared_stuff = (struct shared_use_st *)shared_memory;

    for (i = 0; i < 30; i++)
    {
        sem_p(fmutex);

        // Do write operation
        item = shared_stuff->buffer[shared_stuff->lo];
    (shared_stuff->buffer)[(shared_stuff->lo)]=0;
    (shared_stuff->lo) = ((shared_stuff->lo)+1) % BUFFER_SIZE;
    printf("Removing item %d\n",item);
        sem_v(fmutex);

    printf("Writing %d\n",item);
    sleep(2);
    }

    if (shmdt(shared_memory) == -1) {
          fprintf(stderr, "shmdt failed\n");
    exit(EXIT_FAILURE);
  }
  printf("Finish!\n");
  getchar();
    exit(EXIT_SUCCESS);
}
```

修改init.c文件，使其能初始化reader和writer：

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <sys/shm.h>
#include <sys/msg.h>
#include "myipc.h"

int main(int argc, char *argv[])
{
    // int i,producer_pid,consumer_pid,item,shmid;
  int i,reader_pid,writer_pid,item,shmid;
  // semaphore mutex,empty,full;
  semaphore fmutex = 1;
    semaphore rdcntmutex = 1;
```

```
    union semun sem_union;
  void *shared_memory = (void *)0;
  struct shared_use_st *shared_stuff;

  if ( (fmutex = semget((key_t)KEY_MUTEX,1,0666|IPC_CREAT)) == -1 ) {
    fprintf(stderr,"Failed to create semaphore!");
    exit(EXIT_FAILURE);
  }
  if ( (rdcntmutex = semget((key_t)KEY_EMPTY,1,0666|IPC_CREAT)) == -1
) {
    fprintf(stderr,"Failed to create semaphore!");
    exit(EXIT_FAILURE);
  }
  // if ( (full = semget((key_t)KEY_FULL,1,0666|IPC_CREAT)) == -1 ) {
  //  fprintf(stderr,"Failed to create semaphore!");
  //  exit(EXIT_FAILURE);
  // }
  if ( (shmid = shmget((key_t)KEY_SHM,sizeof(struct
shared_use_st),0666|IPC_CREAT)) == -1 ) {
    fprintf(stderr,"Failed to create shared memory!");
    exit(EXIT_FAILURE);
  }

      sem_union.val = 1;
      if (semctl(fmutex, 0, SETVAL, sem_union) == -1) {
    fprintf(stderr,"Failed to set semaphore!");
    exit(EXIT_FAILURE);
  }

      sem_union.val = 0;
      if (semctl(rdcntmutex, 0, SETVAL, sem_union) == -1) {
    fprintf(stderr,"Failed to set semaphore!");
    exit(EXIT_FAILURE);
  }

    //  sem_union.val = BUFFER_SIZE;
    //  if (semctl(empty, 0, SETVAL, sem_union) == -1) {
  //  fprintf(stderr,"Failed to set semaphore!");
  //  exit(EXIT_FAILURE);
  // }

  if ( (shared_memory = shmat(shmid,(void *)0,0) ) == (void *)-1) {
    fprintf(stderr,"shmat failed\n");
    exit(EXIT_FAILURE);
  }
  shared_stuff = (struct shared_use_st *)shared_memory;

  for(i=0;i<BUFFER_SIZE;i++)
  {
```

```
    shared_stuff->buffer[i] = 0;
  }
  shared_stuff -> lo = 0;
  shared_stuff -> hi = 0;
  shared_stuff -> cur = 0;

    exit(EXIT_SUCCESS);
}
```

修改makefile，使其能编译运行reader和writer：

```
default: myipc init.c producer.c consumer.c
  gcc -o init myipc.o init.c
  gcc -o producer myipc.o producer.c
  gcc -o consumer myipc.o consumer.c
  gcc -o reader myipc.o reader.c
  gcc -o writer myipc.o writer.c
myipc: myipc.c
  gcc -c myipc.c
```

四、测试结果

进入文件目录，编译运行

```
cd /home/yuqianhou/文档/pc_semaphore
make
./init
```

先运行读者：

```
./reader
```

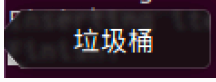
再新建一个shell，运行写者：

```
./writer
```

运行结果如下：

```
yuqianhou@yuqianhou-VirtualBox:~/文档/pc_semaphore$ ./reader
Reading 1
Inserting item 1
Reading 2
Inserting item 2
Reading 3
Inserting item 3
Reading 4
Inserting item 4
Reading 5
Inserting item 5
Reading 6
Inserting item 6
Reading 7
Inserting item 7
Reading 8
Inserting item 8
Reading 9
Inserting item 9
Reading 10
Inserting item 10
Reading 11
Inserting item 11
Reading 12
Inserting item 12
Reading 13
Inserting item 13
Reading 14
Inserting item 14
Reading 15
Inserting item 15
Reading 16
Inserting item 16
Reading 17
Inserting item 17
Reading 18
Inserting item 18
Reading 19
```

```
Reading 20
Inserting item 20
Reading 21
Inserting item 21
Reading 22
Inserting item 22
Reading 23
Inserting item 23
Reading 24
Inserting item 24
Reading 25
Inserting item 25
Reading 26
Inserting item 26
Reading 27
Inserting item 27
Reading 28
Inserting item 28
Reading 29
Inserting item 29
Reading 30
Inserting item 30
```



```
yuqianhou@yuqianhou-VirtualBox:~$ cd /home/yuqianhou/文档/pc_semaphore
yuqianhou@yuqianhou-VirtualBox:~/文档/pc_semaphore$ ./writer
Removing item 11
Writing 11
Removing item 12
Writing 12
Removing item 13
Writing 13
Removing item 14
Writing 14
Removing item 15
Writing 15
Removing item 16
Writing 16
Removing item 27
Writing 27
Removing item 28
Writing 28
Removing item 29
Writing 29
Removing item 30
Writing 30
Removing item 21
Writing 21
Removing item 22
Writing 22
Removing item 23
Writing 23
Removing item 24
Writing 24
Removing item 25
Writing 25
Removing item 26
Writing 26
Removing item 0
Writing 0
Removing item 0
Writing 0
Removing item 0
Writing 0
Removing item 0
Writing 0
```

五、遇到的问题和解决办法

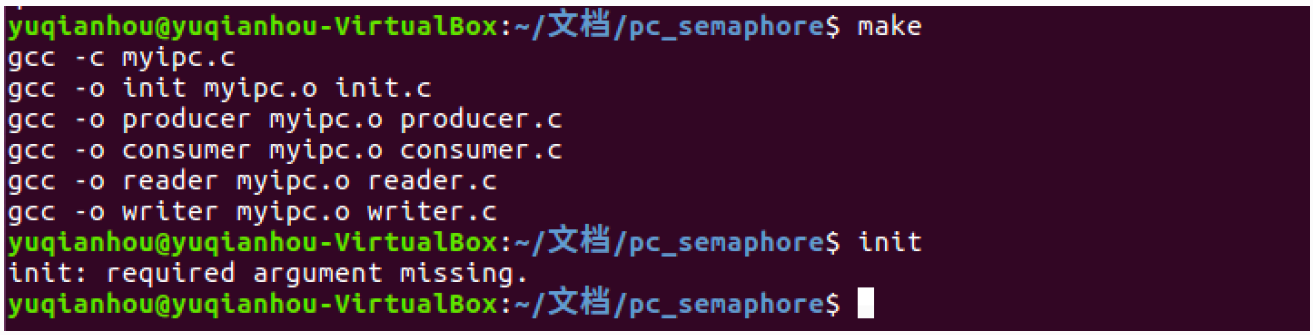
5.1 编译代码问题

最开始编译示例代码时，由于对文件及命令不熟悉，且 `README中` 提到：

```
type "make" to compile

use "init" each time to cleanup
```

因此我误以为直接输入 `make` 和 `init` 两条命令即可运行，导致无法运行出结果：



5.2 修改init文件

编写好代码后，放入Ubuntu中运行时，发现输入 `./reader` 命令后并没有出现结果。于是发现init文件之前是用来初始化 `producer` 和 `consumer` 的。最后将其修改成用来初始化 `reader` 和 `writer` 即可。