进程间同步-生产者消费者问题实验报告

1120213582 陈建腾 网络空间安全学院

1 实验背景

1.1 实验目的

Make C or C++ programs to illustrate the Producer and Consumer synchronization problem. You will have to create several processes to simulate the producers and consumers. Use shared memory to implement the shared buffer among producers and consumers. Use semaphore to synchronize the processes. Here are some constraints for the problem.

1.1.1 Notes

1.A shared buffer with 3 slots, initially are all empty.

2.Two producers
- Randomly wait for a period of time and put product into the buffer.
- Wait if the buffer slots are all full
- Repeat 6 times.

3.Three consumers
- Randomly wait for a period of time and fetch a product from the buffer.
- Wait if the buffer slots are all empty.
- Repeat 4 times.

1.1.2 Hints

- Display the status of the buffer and the time when a product has been put into or removed from the buffer.
- Use process (not thread) to simulate the consumer and producer.
- Use system calls like fork() to create new processes. Use system calls like shmget() to create shared memory and semget() to create semaphores in Linux.
- Implement a Linux version.

1.2 实验环境

- VMWare Workstation 16
- Kali GNU/Linux Rolling
- Linux kali 6.0.0-kali3-amd64
- gcc (Debian 12.2.0-14) 12.2.0

1.3 实验分析

1. 通过宏定义对信号量中的集中索引进行替换，增强可读性并且防止出现逻辑bug
2. 定义缓冲区结构体buffer
3. 通过辅助函数print_Buffer对缓冲区内容及状况进行打印
4. 通过两个循环fork多个多种进程，在子进程函数执行完毕后解除共享段与子进程的连接，并且使用exit()阻断执行
5. 通过随机数确定等待时间

1.3.1 源代码

```
#include <stdio.h>
#include <time.h>
#include <string.h>
```

```c
#include <stdlib.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <sys/wait.h>

#define PRODUCER_NUM 2
#define CONSUMER_NUM 3
#define PRODUCER_REPEAT 6
#define CONSUMER_REPEAT 4
#define OK 1
#define ERROR -1
#define BUFFER_LENGTH 3
#define EMPTY 0
#define FULL 1
#define MUTEX 2

struct buffer
{
    int item[BUFFER_LENGTH];
    int head;
    int tail;
    int empty;
};

void print_Buffer(struct buffer *a)
{
    printf("Current Data:");
    for (int i = a->head;;)
    {
        printf("%d  ", a->item[i]);
        i++;
        i %= 3;
        if (i == a->tail)
        {
            printf("\n\n");
            return;
        }
    }
}

int P(int semid, int n)
{
    if (n < 0)
    {
        printf("index of array can not equals a minus value!\n");
        return ERROR;
    }
    sembuf temp;
    temp.sem_num = n;
    temp.sem_op = -1;
    temp.sem_flg = 0;
    semop(semid, &temp, 1);
    return OK;
}
```

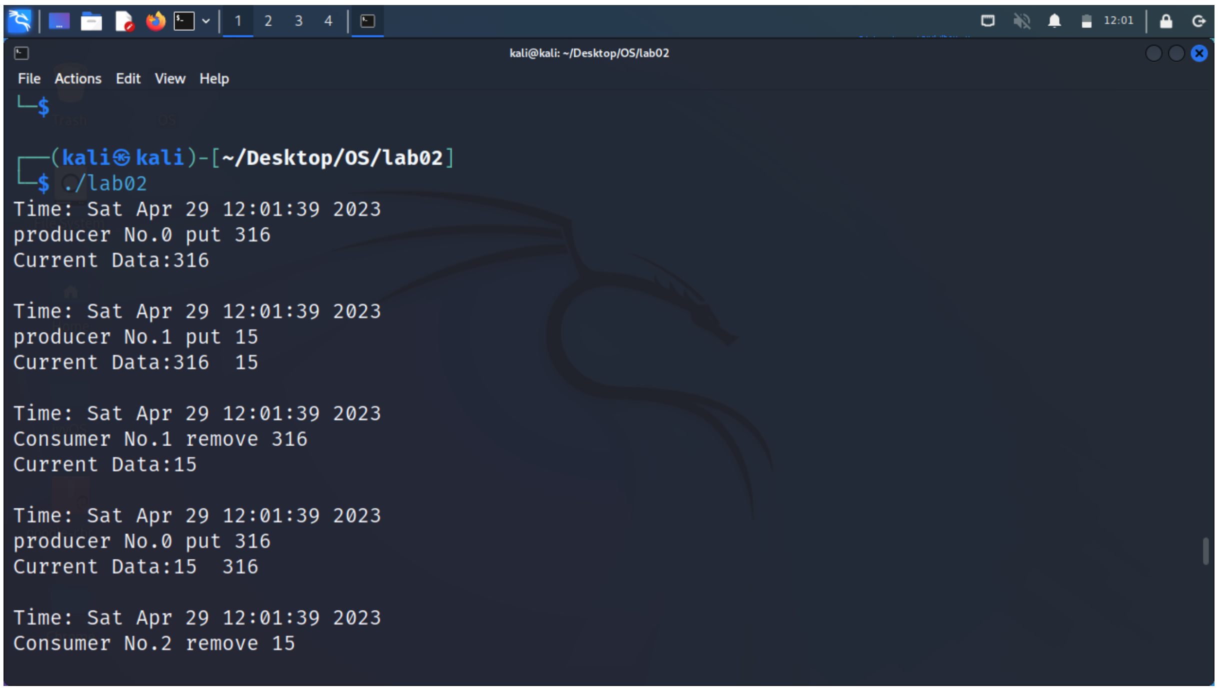
```c
int V(int semid, int n)
{
    if (n < 0)
    {
        printf("index of array can not equals a minus value!\n");
        return ERROR;
    }
    sembuf temp;
    temp.sem_num = n;
    temp.sem_op = 1;
    temp.sem_flg = 0;
    semop(semid, &temp, 1);
    return OK;
}

int main()
{
    int semid = semget(1234, 3, IPC_CREAT | 0600); // create the semaphore with semget
    if (semid < 0)
    {
        printf("semget ERROR\n");
        exit(0);
    }

    // initialize the semaphore, EMPTY->3, FULL ->0, MUTEX->1
    semctl(semid, EMPTY, SETVAL, 3);
    semctl(semid, FULL, SETVAL, 0);
    semctl(semid, MUTEX, SETVAL, 1);

    int shmid = shmget(5678, sizeof(buffer), IPC_CREAT | 0600); // create shared memory
with shmget
    if (shmid < 0)
    {
        printf("shmget ERROR\n");
        exit(0);
    }

    buffer *addr = (buffer *)shmat(shmid, 0, 0);

    if (addr == (void *)-1)
    {
        printf("shmat ERROR\n");
        exit(0);
    }

    addr->head = 0;
    addr->tail = 0;
    addr->empty = 1;

    // PRODUCER
    for (int i = 0; i < PRODUCER_NUM; i++)
    {
        pid_t pid = fork();
        if (pid < 0)
        {
            printf("Producer fork ERROR!\n");
            exit(0);
        }
        if (pid == 0)
        {
            addr = (buffer *)shmat(shmid, 0, 0);
            if (addr == (void *)-1)
            {
                printf("Producer shmat ERROR\n");
                exit(0);
            }

            for (int j = 0; j < PRODUCER_REPEAT; j++)
            {
                srand((unsigned)time(NULL) + getpid());
                sleep(rand() % 1);
                P(semid, EMPTY);
                P(semid, MUTEX);
                int num = rand() % 1000;
                addr->item[addr->tail] = num;
                addr->tail = (addr->tail + 1) % 3;
                addr->empty = 0;

                time_t t;
                time(&t);
                printf("Time: %s", ctime(&t));
                printf("producer No.%d put %d\n", i, num);
                print_Buffer(addr);
                V(semid, FULL);
                V(semid, MUTEX);
            }
            shmdt(addr);
            exit(0);
        }
    }

    // CONSUMER
    for (int i = 0; i < CONSUMER_NUM; i++)
    {
        pid_t pid = fork();
        if (pid < 0)
        {
            printf("Consumer fork ERROR!\n");
            exit(0);
        }
        if (pid == 0)
        {
            addr = (buffer *)shmat(shmid, 0, 0);
            if (addr == (void *)-1)
            {
                printf("Consumer shmat ERROR!\n");
                exit(0);
            }

            for (int j = 0; j < CONSUMER_REPEAT; j++)
            {
                srand((unsigned)time(NULL) + getpid());
                sleep(rand() % 1);
                P(semid, FULL);
                P(semid, MUTEX);
                int num = addr->item[addr->head];
                addr->head = (addr->head + 1) % 3;
                // the buffer is empty or not
                if (addr->head == addr->tail)
                    addr->empty = 1;
                else
                    addr->empty = 0;

                time_t t;
                time(&t);
                printf("Time: %s", ctime(&t));
                printf("Consumer No.%d remove %d\n", i + 1, num);

                if (addr->empty == 0)
                    print_Buffer(addr);
                else
                    printf("the Buffer is empty\n\n");
                V(semid, EMPTY);
                V(semid, MUTEX);
            }
            shmdt(addr);
            exit(0);
        }
    }

    while (wait(0) != -1);
    shmdt(addr);
    semctl(semid, IPC_RMID, 0);
    shmctl(shmid, IPC_RMID, 0);
    return 0;
}
```

2 运行与结果

2.1 运行

1. gcc lab02.cpp -o lab02
2. ./lab02

2.2 结果



```
└─$

┌──(kali㉿kali)-[~/Desktop/OS/lab02]
└─$ ./lab02
Time: Sat Apr 29 12:01:39 2023
producer No.0 put 316
Current Data:316

Time: Sat Apr 29 12:01:39 2023
producer No.1 put 15
Current Data:316  15

Time: Sat Apr 29 12:01:39 2023
Consumer No.1 remove 316
Current Data:15

Time: Sat Apr 29 12:01:39 2023
producer No.0 put 316
Current Data:15  316

Time: Sat Apr 29 12:01:39 2023
Consumer No.2 remove 15
```

```
Time: Sat Apr 29 12:07:03 2023
producer No.1 put 234
Current Data:234

Time: Sat Apr 29 12:07:03 2023
producer No.0 put 235
Current Data:234  235

Time: Sat Apr 29 12:07:03 2023
Consumer No.1 remove 234
Current Data:235

Time: Sat Apr 29 12:07:03 2023
producer No.1 put 234
Current Data:235  234

Time: Sat Apr 29 12:07:03 2023
Consumer No.3 remove 235
Current Data:234

Time: Sat Apr 29 12:07:03 2023
producer No.0 put 235
Current Data:234  235

Time: Sat Apr 29 12:07:03 2023
Consumer No.2 remove 234
Current Data:235

Time: Sat Apr 29 12:07:03 2023
Consumer No.1 remove 235
the Buffer is empty

Time: Sat Apr 29 12:07:03 2023
producer No.1 put 234
Current Data:234
```

```
Time: Sat Apr 29 12:07:03 2023
producer No.0 put 235
Current Data:234  235

Time: Sat Apr 29 12:07:03 2023
Consumer No.1 remove 234
Current Data:235

Time: Sat Apr 29 12:07:03 2023
producer No.0 put 235
Current Data:235  235

Time: Sat Apr 29 12:07:03 2023
Consumer No.3 remove 235
Current Data:235

Time: Sat Apr 29 12:07:03 2023
producer No.1 put 234
Current Data:235  234

Time: Sat Apr 29 12:07:03 2023
Consumer No.2 remove 235
Current Data:234

Time: Sat Apr 29 12:07:03 2023
Consumer No.1 remove 234
the Buffer is empty

Time: Sat Apr 29 12:07:03 2023
producer No.0 put 235
Current Data:235

Time: Sat Apr 29 12:07:03 2023
producer No.1 put 234
Current Data:235  234

Time: Sat Apr 29 12:07:03 2023
Consumer No.3 remove 235
Current Data:234

Time: Sat Apr 29 12:07:03 2023
producer No.0 put 235
Current Data:234  235

Time: Sat Apr 29 12:07:03 2023
Consumer No.2 remove 234
Current Data:235

Time: Sat Apr 29 12:07:03 2023
producer No.1 put 234
Current Data:235  234

Time: Sat Apr 29 12:07:03 2023
Consumer No.3 remove 235
Current Data:234
```

```
Time: Sat Apr 29 12:07:03 2023
Consumer No.2 remove 234
the Buffer is empty
```