OS project four 实验报告
——Producer-Consumer Problem

5140309201

黄晟

一、 实验要求

通过程序模拟生产者—消费者问题，利用信号量及互斥锁来防止冲突。

二、 实现过程

1、定义buffer_item及缓冲区大小；

```
typedef int buffer_item;
#define BUFFER_SIZE 5
```

2、编写函数insert_item()以及remove_item()来判断是否成功插入或者删除缓冲区内容；

```
int insert_item(buffer_item item)
{
    if (count < BUFFER_SIZE){
        buffer[count++] = item;
        return 0;
    }
    else return -1;
}

int remove_item(buffer_item *item)
{
    if (count > 0){
        *item = buffer[--count];
        return 0;
    }
    else return -1;
}
```

3、编写主函数，完成初始化、创建进程以及结束的功能；

```
int main(int argc, char *argv[])
{
    if (argc < 4){
        printf("error input format\n");
    }

    int sleepTime = atoi(argv[1]);
    int numPro = atoi(argv[2]);
    int numCon = atoi(argv[3]);

    pthread_mutex_init(&mutex, NULL);

    sem_init(&empty, 0, BUFFER_SIZE);
    sem_init(&full, 0, 0);

    pthread_attr_init(&attr);

    printf("producer-consumer problem starts:\n");

    int i;
    for(i = 0; i < numPro; ++i){
        pthread_create(&tid, &attr, producer, NULL);
    }

    for(i = 0; i < numCon; ++i){
        pthread_create(&tid, &attr, consumer, NULL);
    }

    sleep(sleepTime);
    exit(0);
```

4、编写producer、consumer函数，注意到这两个函数在critical section外要共用一对互斥锁mutex，然后在其外部再套一层信号量，以判断当前是否能执行插入或者删除操作。要注意到producer是wait empty并post full，而consumer是wait full并post empty。

```c
void *producer(void *param)
{
    buffer_item randNum;

    while(1){
        int r = rand()%rand_divisor;
        sleep(r);

        randNum = rand();

        sem_wait(&empty);
        pthread_mutex_lock(&mutex);

        printf("Producer produced %d\n", randNum);
        if (insert_item(randNum))
            fprintf(stderr, "report error condition\n");

        pthread_mutex_unlock(&mutex);
        sem_post(&full);
    }
}
```

Figure 1 producer 函数

```c
void *consumer(void *param)
{
    buffer_item randNum;

    while(1){
        int r = rand()%rand_divisor;
        sleep(r);

        sem_wait(&full);
        pthread_mutex_lock(&mutex);

        if (remove_item(&randNum))
            fprintf(stderr, "report error condition\n");
        else
            printf("Consumer consumed %d\n", randNum);

        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
}
```

Figure 2 consumer 函数

三、 实现效果

1、当有10个生产者线程、10个消费者线程时，运行结果如图；

```
Consumer consumed 1129566413
Producer produced 184803526
Consumer consumed 184803526
Producer produced 137806862
Producer produced 982906996
Consumer consumed 982906996
Consumer consumed 137806862
Producer produced 1937477084
Producer produced 572660336
Producer produced 1159126505
Consumer consumed 1159126505
Consumer consumed 572660336
Producer produced 1100661313
Consumer consumed 1100661313
```

2、当有10个生产者线程、1个消费者线程时，运行结果如图；

```
cyril@ubuntu:~/Desktop/project/4$ ./a.out 10 10 1
producer-consumer problem starts:
Producer produced 1350490027
Producer produced 1102520059
Producer produced 1967513926
Producer produced 1365180540
Producer produced 1303455736
Consumer consumed 1303455736
Producer produced 294702567
```

四、 心得与体会

1、本来在线程create后加了一句pthread_join，这就导致了程序不停地运行，这是因为pthread_join是阻塞并等待进程结束，而生产者、消费者进程却没有有效的退出命令，因此线程会不断的执行。这就导致最后main函数里的sleep和exit没有执行。而把这句话去掉以后程序就正常运行了。

最后衷心感谢吴老师和助教在我做 project 的过程中给予我的指导和帮助。