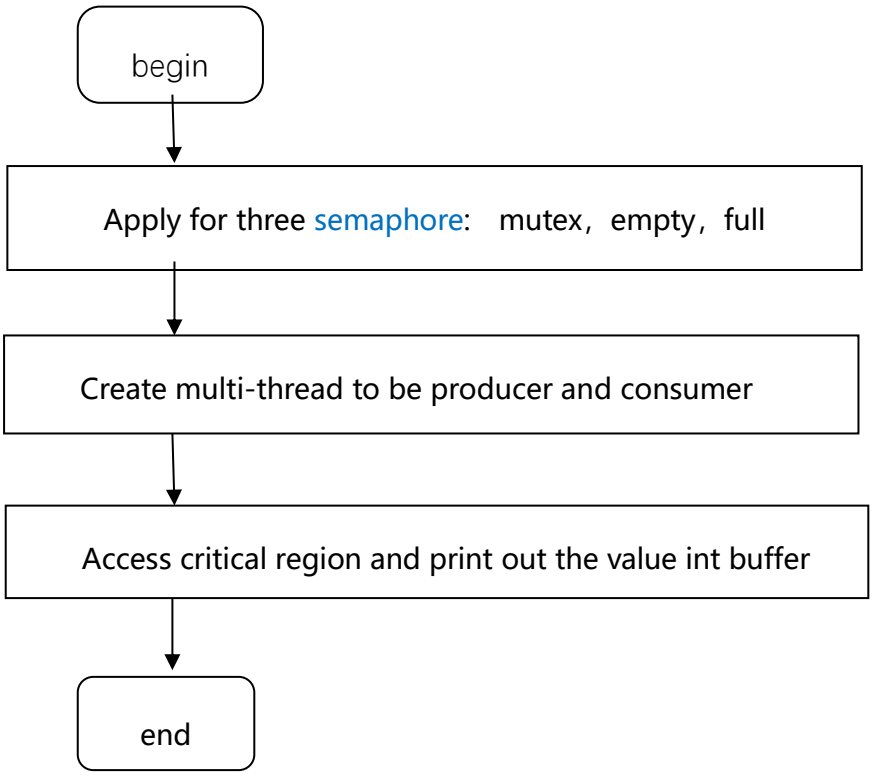
Operating system experiment report

1. **Name:** Luoyu Mei **Number:** 71117408 **Date:** 05/05/2019
2. **Working target:** Handle consumer producer problem in Windows and Linux operating system by using Windows API and PThread API.

   Working environment: Window10 as basic operating system with "virtual box" virtual machine and Ubuntu18.04 running on it. Using GCC version 8.2.0 on Windows and version 7.4.0 on Ubuntu.
3. **Steps:**
   1. In order to make the program satisfy "Critical Section", I set three semaphore semaphore mutex = 1, empty = 10, full = 0 and a char buffer array char buffer[10] to be the critical region.
   2. Producer produce into buffer using signal and make buffer turn to 'A'. Consumer consume buffer by changing it to 'B'. Either producer or consumer access "Critical region", I print buffer out to show changes.
   3. Create multi-thread which include 5 producer and 5 consumer, the maximum frequency for each thread to access "Critical region" is 10.
4. **Flow chart:**



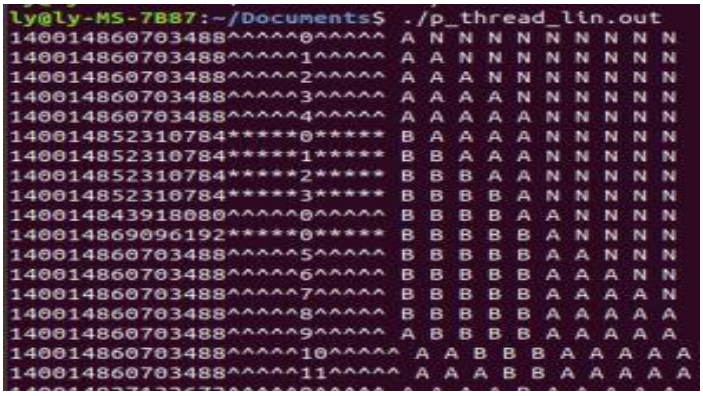
5. Main data structure:

I use three semaphores, counter semaphore empty and full, which were respectively set initial value 10 and 0, exclusive semaphore mutex to control the process accession. I define a buffer in type of char to save the status. Whenever consumer or producer access the buffer, I change the value in it and print it out.

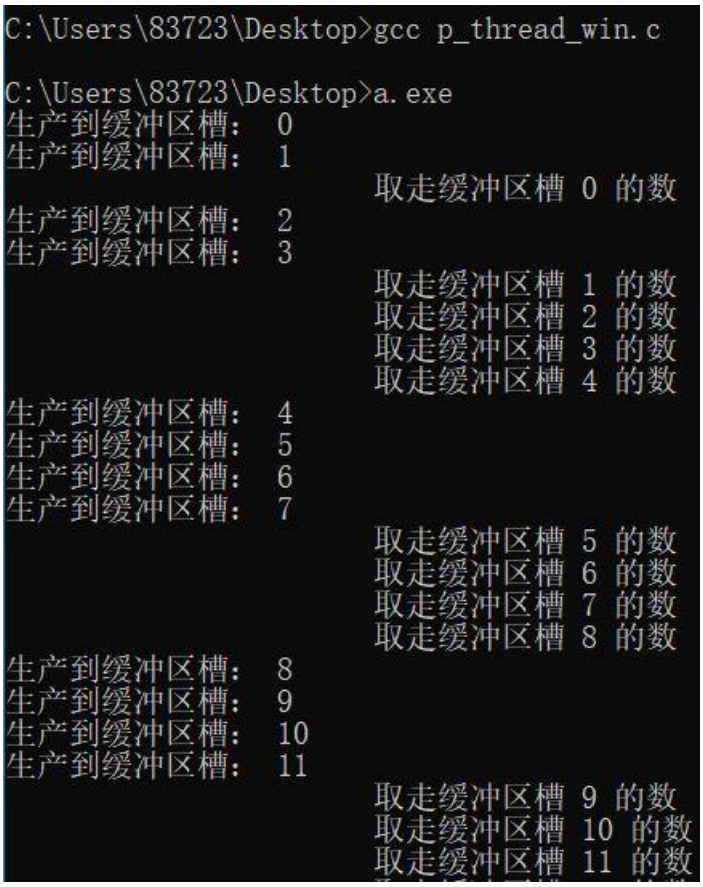
The code file of my program will also be bale together with this report.

6. Experiment result:

Ubuntu:



Windows:



7. Filling of experiment:

It isn't difficult to design my program, however the realization of it in Windows operating system maybe difficult for me as a beginner. Windows API is power but difficult to learn, I spend half of my coding time to search for reference of it. The most acquisition I got in this experiment is the using of Windows API to comply multi thread.

8. Code for Linux (如果您在看 PDF，代码会在下一页出现):

```c
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>

#define BUFFER_SIZE 10//缓冲区大小为 10
char *buffer;
sem_t mutex,empty,full;//三个信号量，互斥信号量 mutex，计数信号量 empty 和 full
int x,y;//生产者和消费者在 buffer 中下标
void output()//输出 buffer 数组
{
    int i;
    for(i=0;i<BUFFER_SIZE;i++)
    {
        printf("%c",buffer[i]);
        printf(" ");
    }
    printf("\n");
}
void *produce()//生产者函数
{
    int j;
    j=0;
    do
    {
        sem_wait(&empty);//buffer 有空余部分，可以生产，并减一
        sem_wait(&mutex);//形成互斥访问，只能一个线程生产
        printf("%lu%s%d%s",pthread_self(),"^^^^^^",j,"^^^^^^ ");//输出当前线程的 id 号，以及正在执行的次数
        buffer[(x++)%BUFFER_SIZE]='A';//生产就赋值 A
        output();//输出 buffer
        j++;
        sem_post(&mutex);//取消互斥
        sem_post(&full);//生成完毕，增加一个可以消费量。
    }while (j!=30);//每个线程可以做 30 次
}
void *consume()//消费者函数
{
    int j;
    j=0;
    do
    {
        sem_wait(&full);//可以消费的量减一
        sem_wait(&mutex);//互斥访问，只能一个线程消费
        printf("%lu%s%d%s",pthread_self(),"*****",j,"***** ");
```

Code for Windows (如果您在看 PDF，代码会在下一页出现):

```c
#include <Windows.h>
#include <stdio.h>
#define N 100
#define TRUE 1
typedef int Semaphore;
Semaphore full = 0, Empty = N;              //共享资源区满槽数目和空槽数目
int in = 0, out = 0;                        //缓冲区生产，消费数据指针
HANDLE mutex;
int ProducerThread[5];
int ConsumerThread[5];
int Buffer[N+4];                            //缓冲区

int produce_item() {                        //生产(随机数)
    return (rand()%N + N)%N;
}

int insert_item(int item) {                 //插入资源
    in %= N;
    printf("生产到缓冲区槽： %d\n",in);
    Buffer[in] = item;
    return Buffer[in++];
}

int remove_item() {                          //移出资源
    out %= N;
    printf("                        取走缓冲区槽 %d 的数\n",out);
    return Buffer[out++];
}

int consume_item(int item) {
    //consume it
}

void down(HANDLE handle) {                   //wait / P
    WaitForSingleObject(handle, INFINITE);
}

void up(HANDLE handle) {                     //signal / V
    ReleaseSemaphore(handle, 1, NULL);
}

DWORD WINAPI producer(LPVOID v) {

    int item;
```