**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №6

по дисциплине

«Операционные системы»

Вариант 2

Студент

Группа АИ-20-1 \_\_\_\_\_\_\_\_\_\_ Глубоков Г.В.

Подпись, дата

Руководитель

к.т.н.,доц. Батищев Р. В.

Подпись, дата

Липецк 2022г.

# Задание кафедры:

Реализовать с помощью семафоров решение задачи «производитель-потребитель», которая сформулирована в следующем виде: написать программу-клиент – потребитель данных из кольцевого буфера и программу-сервер – генератор (производитель) данных в буфер. В кольцевом буфере можно выделить первую занятую, последнюю занятую, далее – первую свободную и затем – последнюю свободную позиции. Сервер генерирует заданное количество данных и после получения доступа к буферу помещает их в буфер, начиная с первой свободной позиции, если это возможно. В противном случае он ждет, когда освободится место в буфере. Клиент забирает заданное количество данных из буфера, освобождая соответствующие позиции в буфере, начиная с первой занятой, если это возможно. В противном случае клиент ждет, когда сервер поместит достаточное количество данных в буфер (количество данных M, генерируемых сервером, количество данных N, считываемых клиентом, размер буфера K заданы в табл. 5 прил.).

|  |  |
| --- | --- |
| № вар. | M, N,K |
| 2 | 5, 20, 50 |

# Теоретические сведения:

# Для работы с именованными семафорами предусмотрены следующие функции:

# sem\_open() — открывает или создает новый семафор, инициализирует его (если она его создала) и возвращает дескриптор, который можно использовать в дальнейшем;

# sem\_post(sem) и sem\_wait(sem) — соответственно инкрементируют и декрементируют значение семафора;

# sem\_getvalue() — возвращает текущее значение семафора;

# sem\_close() — закрывает семафор, ранее открытый вызывающим процессом;

# sem\_unlink() — удаляет имя семафора и делает его кандидатом на удаление; само удаление произойдет, когда семафор закроет все процессы.

# Ход работы:

## Код программы на C++:

#include <iostream>

#include <pthread.h>

#include <semaphore.h>

using namespace std;

int G\_D=5;

int R\_D=20;

sem\_t sem\_rec\_data;

sem\_t sem\_post\_data;

pthread\_t server;

pthread\_t client;

pthread\_mutex\_t mutx = PTHREAD\_MUTEX\_INITIALIZER;

struct Buffer{

int data[50] = {0};

int inputPoint = 0;

int outputPoint = 0;

};

void printBuffer(Buffer \* buffer){

for(int value: buffer->data){

if(value!=0)

cout << value << " " ;

else

cout << value << " " ;

}

}

static void\* serverThread(void\* buffer){

Buffer \* buf = (Buffer \*) buffer;

for(int j = 0 ; j < 10 ; j++){

for(int i = 0 ; i < G\_D ; i++){

if(i<=0){

sem\_wait(&sem\_rec\_data);

} else {

sem\_trywait(&sem\_rec\_data);

}

if(buf->inputPoint==50){

buf->inputPoint=0;

}

buf->data[buf->inputPoint] = rand() % 9 + 1;

buf->inputPoint++;

}

pthread\_mutex\_lock(&mutx);

cout << "Server" << endl;

printBuffer(buf);

cout << endl;

pthread\_mutex\_unlock(&mutx);

// if(j==9){

// for(int i=0 ; i < G\_D+10 ; i++){

// sem\_post(&sem\_post\_data);

// }

for(int i=0 ; i < G\_D ; i++){

sem\_post(&sem\_post\_data);

}

}

for(int i=0 ; i < 10 ; i++){

sem\_post(&sem\_post\_data);

}

pthread\_exit(NULL);

}

static void \* clientThread(void\* buffer){

Buffer \* buf = (Buffer \*) buffer;

for(int j = 0 ; j < 3 ; j++){

for(int i = 0 ; i < R\_D ; i++){

sem\_wait(&sem\_post\_data);

if(buf->outputPoint==50){

buf->outputPoint=0;

}

buf->data[buf->outputPoint] = 0;

buf->outputPoint++;

}

pthread\_mutex\_lock(&mutx);

cout << "Client" << endl;

printBuffer(buf);

cout << endl;

pthread\_mutex\_unlock(&mutx);

int currentValueRD;

sem\_getvalue(&sem\_rec\_data,&currentValueRD);

if(currentValueRD < R\_D){

int sizeOfReqiuredData = R\_D + currentValueRD;

for(int i=0 ; i < sizeOfReqiuredData ; i++){

sem\_post(&sem\_rec\_data);

}

}

}

pthread\_exit(NULL);

}

int main(int argc, char \*argv[]) {

srand(time(NULL));

Buffer buffer;

sem\_init(&sem\_rec\_data,0,R\_D);

sem\_init(&sem\_post\_data,0,0);

pthread\_create(&client,NULL,&clientThread,&buffer);

pthread\_create(&server,NULL,&serverThread,&buffer);

pthread\_join(server,NULL);

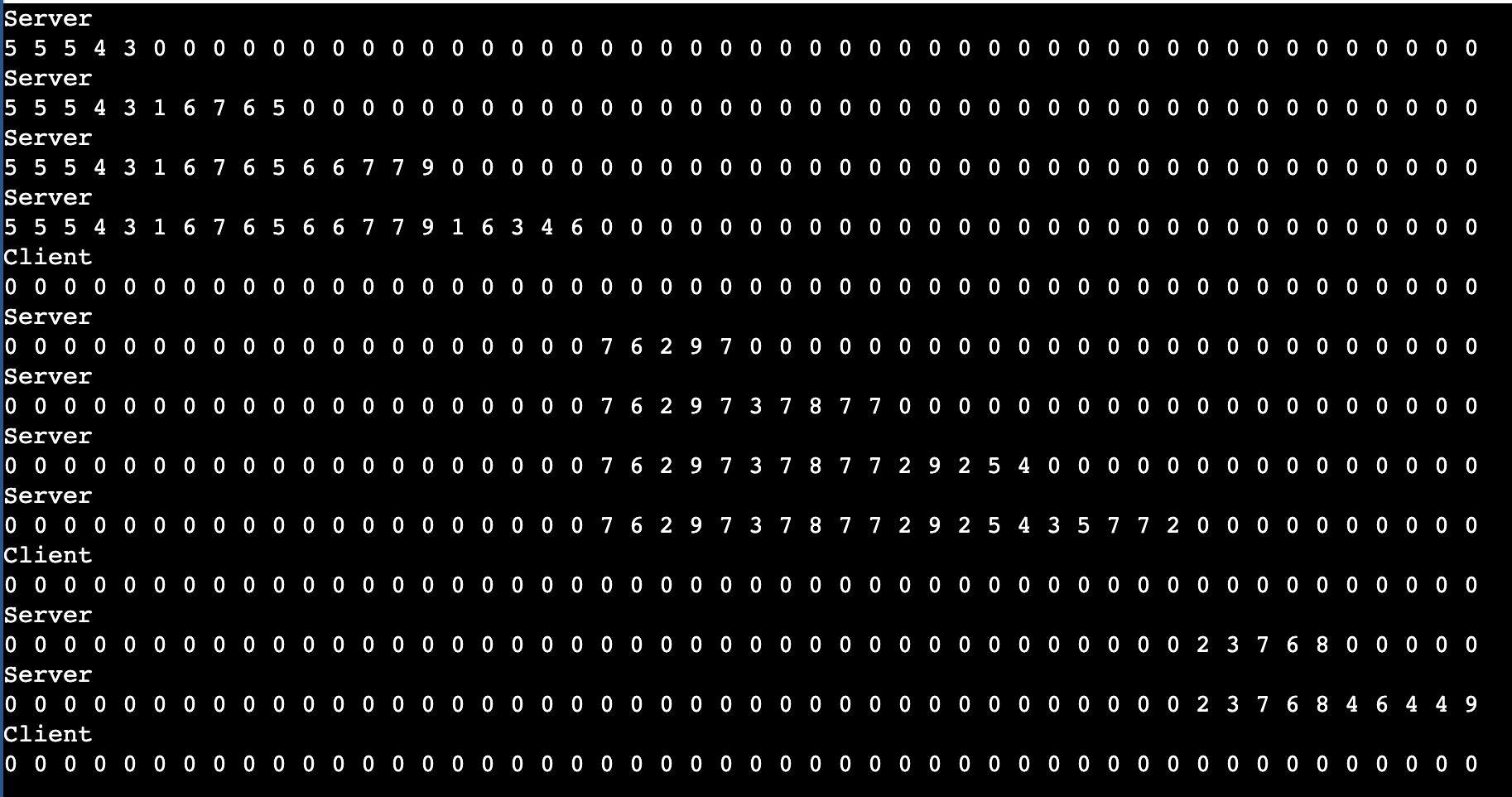
pthread\_join(client,NULL);

sem\_destroy(&sem\_rec\_data);

sem\_destroy(&sem\_post\_data);

return 0;

}



В ходе выполнения лабораторной работы были получены основные навыки работы с семафорами Linux API