МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной Техники

ОТЧЕТ

по лабораторной работе №9

по дисциплине «Организация процессов и программирование в среде Linux»

ТЕМА: «ОБМЕН ДАННЫМИ ЧЕРЕЗ РАЗДЕЛЯЕМУЮ ПАМЯТЬ»

Студент гр. 8308 _____ Тайсумов И.И.

Преподаватель _____ Разумовский Г.В.

Санкт-Петербург

2021

Цель работы.

Целью лабораторной работы является знакомство с организацией разделяемой памяти и системными функциями, обеспечивающими обмен данными между процессами.

Задание.

Написать 3 программы, которые запускаются в произвольном порядке и построчно записывают свои индивидуальные данные в один файл через определенный промежуток времени. Пока не закончит писать строку одна программа, другие две не должны обращаться к файлу. Частота записи данных в файл и количество записываемых строк определяются входными параметрами, задаваемыми при запуске каждой программы. При завершении работы одной из программ другие должны продолжить свою работу. Синхронизация работы программ должна осуществляться с помощью общих переменных, размещенных в разделяемой памяти.

Обработка результатов эксперимента.

Программы были разработаны и скомпилированы. Результат их работы представлен на следующих рисунках:

```
^[[Ataisumov@taisumov-TM1703:~/CLionProjects/CPP_lab9$ ./second 5
Launch first program to continue, waiting...
recieved        [3 6 17 15 13 15 6 12 9 1]
sent:           [2 5 16 14 12 14 5 11 8 0]

recieved        [2 5 16 14 12 14 5 11 8 0]
sent:           [1 4 15 13 11 13 4 10 7 -1]

recieved        [1 4 15 13 11 13 4 10 7 -1]
sent:           [0 3 14 12 10 12 3 9 6 -2]

recieved        [0 3 14 12 10 12 3 9 6 -2]
sent:           [-1 2 13 11 9 11 2 8 5 -3]

recieved        [-1 2 13 11 9 11 2 8 5 -3]
sent:           [-2 1 12 10 8 10 1 7 4 -4]

The work is done!
taisumov@taisumov-TM1703:~/CLionProjects/CPP_lab9$
```

Рисунок 1. Работа первой программы

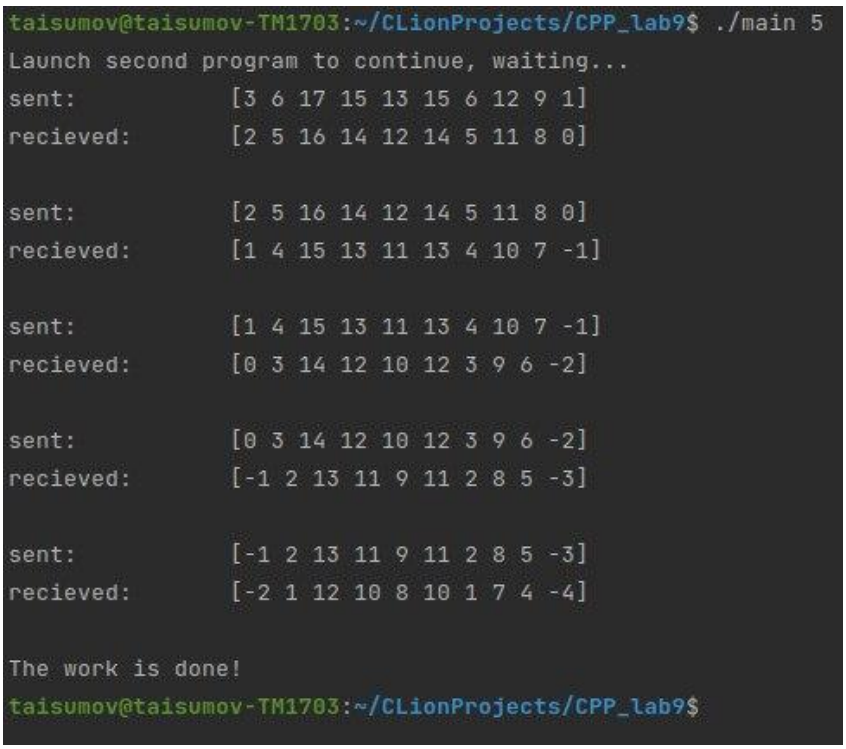

Рисунок 2. Работа второй программы

Текст программы приведен в приложении.

Вывод.

При выполнении лабораторной работы изучены и использованы организация разделяемой памяти и системные функции, обеспечивающие обмен данными между процессами. Программа, разработанная в соответствии с заданием, работает корректно.

ПРИЛОЖЕНИЕ

main.cpp:

```
/**
 * Ислам Тайсумов, группа 8308
 *
 * Компиляция программы:
 * 1. g++ -o main main.cpp
 * 2. ./main x (x - количество итераций)
 *
**/

#include <iostream>
#include <sys/shm.h>
#include <unistd.h>

using namespace std;

void printArray(int *arr, const size_t size) {
   cout << "[";
   for(int i = 1; i < size; ++i) {
      cout << arr[i];
      //cout << (i < size - 1) ? " " : "";
      if(i < size-1)
         cout << " ";
   }
   std::cout << "]\n";
}

int main(int arc, char* argv[]) {
   const size_t BUFFERSIZE = 11;
   const int firstKey = 1111;
   const int secondKey = 2222;
   int numOfIterations = atoi(argv[1]);

   // shmget - присваивает идентификатор разделяемому сегменту памяти
   // IPC_CREAT служит для создания нового сегмента. 0666 - на чтение и запись
   int firstMemorySegment = shmget(firstKey,
                           sizeof(int) * BUFFERSIZE,
                           0666 | IPC_CREAT);

   if(firstMemorySegment == -1) {
      cout << "Error in shmget()!" << endl;
      return -1;
   }

   // функция shmat подстыковывает сегмент разделяемой памяти
   // shmid к адресному пространству вызывающего процесса
   int *sendArray = (int*)shmat(firstMemorySegment,
                        nullptr,
```

```
                    0);

if(sendArray == nullptr) {
   cout << "Error in shmat()!" << endl;
   return -1;
}

// дожидаемся открытия второй программы
cout << "Launch second program to continue, waiting..." << endl;
int secondMemorySegment;
do {
   secondMemorySegment = shmget(secondKey,
                        sizeof(int) * BUFFERSIZE,
                        0666);
} while(secondMemorySegment == -1);

// функция shmat подстыковывает сегмент разделяемой памяти
// shmid к адресному пространству вызывающего процесса
int *receiveArray = (int*)shmat(secondMemorySegment,
                        nullptr,
                        0);

if(receiveArray == nullptr) {
   cout << "Error in shmat()!" << endl;
   return -1;
}

// заполняем массив случайными числами
for(int i = 1; i < BUFFERSIZE; ++i) {
   sendArray[i] = rand() % 20;
}

while (numOfIterations--) {
   // отправка данных
   while(sendArray[0] == 1) {}
   cout << "sent:\t\t";
   printArray(sendArray, BUFFERSIZE);
   sendArray[0] = 1;

   // прием данных
   while(receiveArray[0] == 0) {}
   for(int i = 1; i < BUFFERSIZE; ++i)
      sendArray[i] = receiveArray[i];
   cout << "recieved:\t";
   printArray(receiveArray, BUFFERSIZE);
   receiveArray[0] = 0;
   cout << endl;
}

cout << "The work is done!" << endl;
```

```
    shmdt(sendArray);
    shmdt(receiveArray);
    shmctl(firstMemorySegment, IPC_RMID, 0);
}
```

second.cpp:

```
/**
 * Ислам Тайсумов, группа 8308
 *
 * Компиляция программы:
 * 1. g++ -o second second.cpp
 * 2. ./second x (x - количество итераций)
 *
**/

#include <iostream>
#include <sys/shm.h>
#include <unistd.h>

using namespace std;

void printArray(int *arr, const size_t size) {
    cout << "[";
    for(int i = 1; i < size; ++i) {
        cout << arr[i];
        //cout << (i < size - 1) ? " " : "";
        if(i < size-1)
            cout << " ";
    }
    cout << "]\n";
}

int main(int arc, char* argv[]) {
    const size_t BUFFERSIZE = 11;
    const int firstKey = 1111;
    const int secondKey = 2222;
    int numOfIterations = atoi(argv[1]);

    // shmget - присваивает идентификатор разделяемому сегменту памяти
    // IPC_CREAT служит для создания нового сегмента. 0666 - на чтение и запись
    int secondMemorySegment = shmget(secondKey,
                sizeof(int)*BUFFERSIZE,
                0666 | IPC_CREAT);

    if(secondMemorySegment == -1){
        cout << "Error in shmget()!" << endl;
        return -1;
```

```
}

// функция shmat подстыковывает сегмент разделяемой памяти
// shmid к адресному пространству вызывающего процесса
int *sendArray = (int*)shmat(secondMemorySegment,
                    nullptr,
                    0);

if(sendArray == nullptr) {
   cout << "Error in shmat()!" << endl;
   return -1;
}

// дожидаемся открытия первой программы
cout << "Launch first program to continue, waiting..." << endl;
int firstMemorySegment;
do {
   firstMemorySegment = shmget(firstKey,
            sizeof(int)*BUFFERSIZE,
            0666);
} while(firstMemorySegment == -1);

// функция shmat подстыковывает сегмент разделяемой памяти
// shmid к адресному пространству вызывающего процесса
int *receiveArray = (int*)shmat(firstMemorySegment,
                    nullptr,
                    0);

if(receiveArray == nullptr) {
   cout << "Error in shmat()!" << endl;
   return -1;
}

while (numOfIterations--) {
   // прием данных
   while (receiveArray[0] == 0) {}
   for(int i = 1; i < BUFFERSIZE; ++i)
      sendArray[i] = receiveArray[i]-1;
   cout << "recieved\t";
   printArray(receiveArray, BUFFERSIZE);
   receiveArray[0] = 0;

   // отправка данных
   while(sendArray[0] == 1) {}
   cout << "sent:\t\t";
   printArray(sendArray, BUFFERSIZE);
   sendArray[0] = 1;
   cout << endl;
}

cout << "The work is done!" << endl;
```

```
    shmdt(sendArray);
    shmdt(receiveArray);
    shmctl(secondMemorySegment, IPC_RMID, 0);
}
```