

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

Студенты гр. 4306

Веретенников Л.М.

Преподаватель

Разумовский Г.В.

Санкт-Петербург
2017

Цель работы

Знакомство с организацией разделяемой памяти и системными функциями, обеспечивающими обмен данными между процессами.

Задание 1. Напишите 3 программы, которые запускаются в произвольном порядке и построчно записывают свои индивидуальные данные в один файл. Пока не закончит писать строку одна программа, другие две не должны обращаться к файлу. Порядок 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.

Откомпилируйте 3 программы и запустите их несколько раз на разных терминалах, так чтобы они писали в файл различное число строк и менялся порядок следования строк.

Текст программы main.cpp:

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

using namespace std;

int main(int argc, char* argv[]) {
    if (argc < 3) {
        cout << "Ошибка. Необходимо указать номер программы и количество строк для вывода" << endl;
        return 1;
    }

    int *array;
    int shmid;

    shmid = shmget(1, 5*sizeof(int), 0666 | IPC_CREAT);
    cout << "shmid: " << shmid << endl;
    if (shmid == -1) {
        cout << "Ошибка. Не удалось создать разделяемый сегмент памяти" << endl;
        return 1;
    }
```

```

    array = (int*)shmat(shmid, nullptr, 0);
    cout << "lock: " << (int*)array << endl;
    if (array == (int*) (-1)) {
        cout << "Ошибка. Не удалось отобразить разделяемую память в адресное
пространство процесса" << endl;
        return 1;
    }

    int numProgram = atoi(argv[1]);
    int countStr = atoi(argv[2]);

    ofstream fout;
    for (int iterations = 0; iterations < countStr; iterations++) {
        array[numProgram] = 1;
        array[0] = numProgram;
        for (int i = 1; i < 4; i++)
            while (i != numProgram && array[i] == 1 && array[0] !=
numProgram);
        fout.open("output.txt", ios_base::app);
        fout << "PID: " << getpid() << endl;
        cout << "PID: " << getpid() << endl;
        fout.close();
        sleep(1);
        array[numProgram] = 0;
    }

    shmdt(array);
    return 0;
}

```

Запуск программы:

PID: 26437
PID: 26436
PID: 26437

Задание 2. Напишите две программы, которые работают параллельно и обмениваются массивом целых чисел через две общие разделяемые области. Через первую область первая программа передает массив второй программе. Через вторую область вторая программа возвращает первой программе массив, каждый элемент которого уменьшен на 1. Обе программы должны вывести получаемую последовательность чисел. Синхронизация работы программ должна осуществляться с помощью общих переменных, размещенных в разделяемой памяти.

Откомпилируйте 2 программы и запустите их на разных терминалах.

Текст программы main.cpp:

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

using namespace std;

int main(int argc, char* argv[]) {
    if (argc < 2) {
        cout << "Ошибка. Программа должна принимать количество итераций" <<
endl;
        return 1;
    }

    int* array_send;
    int* array_receive;

    int shmid;
    int shmid_another;

    //SEND ARRAY
    shmid = shmget(2, 6*sizeof(int), 0666 | IPC_CREAT);
    if (shmid == -1) {
        cout << "Ошибка. Не удалось создать разделяемый сегмент памяти" <<
endl;
        return 1;
    }

    array_send = (int*)shmat(shmid, nullptr, 0);
    if (array_send == (int*) (-1)) {
        cout << "Ошибка. Не удалось отобразить разделяемую память в адресное
пространство процесса" << endl;
        return 1;
    }
}
```

```

//RECEIVE ARRAY
shmctl(shmid, IPC_RMID, nullptr);
shmctl(shmid_another, IPC_RMID, nullptr);
shmctl(shmid_another, IPC_CREAT);
if (shmctl(shmid_another, IPC_CREAT) == -1) {
    cout << "Ошибка. Не удалось создать разделяемый сегмент памяти" <<
endl;
    return 1;
}

array_receive = (int*)shmat(shmid_another, nullptr, 0);
if (array_receive == (int*) (-1)) {
    cout << "Ошибка. Не удалось отобразить разделяемую память в адресное
пространство процесса" << endl;
    return 1;
}

srand(time(nullptr));
for (int i = 1; i <= 5; i++)
    array_receive[i] = rand() % 10;

int iterations = atoi(argv[1]);
while (iterations != 0) {
    cout << "ARRAY_SEND: [ ";
    for (int i = 1; i <= 5; i++) {
        array_send[i] = array_receive[i];
        cout << array_send[i] << " ";
    }
    cout << "]" << endl;
    array_send[0] = 1;
    sleep(1);

    while (array_receive[0] != 1);
    cout << "ARRAY_RECEIVE: [ ";
    for (int i = 1; i < 6; i++) {
        cout << array_receive[i] << " ";
    }
    cout << "]" << endl;
    array_receive[0] = 0;
    sleep(1);

    iterations--;
}
array_send[0] = -1;

shmctl(shmid, IPC_RMID, nullptr);
shmctl(shmid_another, IPC_RMID, nullptr);
return 0;
}

```

Текст программы lab9_client.cpp:

```

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

```

```

using namespace std;

```

```

int main() {

```

```

int* array_send;
int* array_receive;

int shmid;
int shmid_another;

//SEND ARRAY
shmid = shmget(3, 6*sizeof(int), 0666 | IPC_CREAT);
if (shmid == -1) {
    cout << "Ошибка. Не удалось создать разделяемый сегмент памяти" <<
endl;
    return 1;
}

array_send = (int*)shmat(shmid, nullptr, 0);
if (array_send == (int*) (-1)) {
    cout << "Ошибка. Не удалось отобразить разделяемую память в адресное
пространство процесса" << endl;
    return 1;
}

//RECEIVE ARRAY
shmid_another = shmget(2, 6*sizeof(int), 0666 | IPC_CREAT);
if (shmid_another == -1) {
    cout << "Ошибка. Не удалось создать разделяемый сегмент памяти" <<
endl;
    return 1;
}

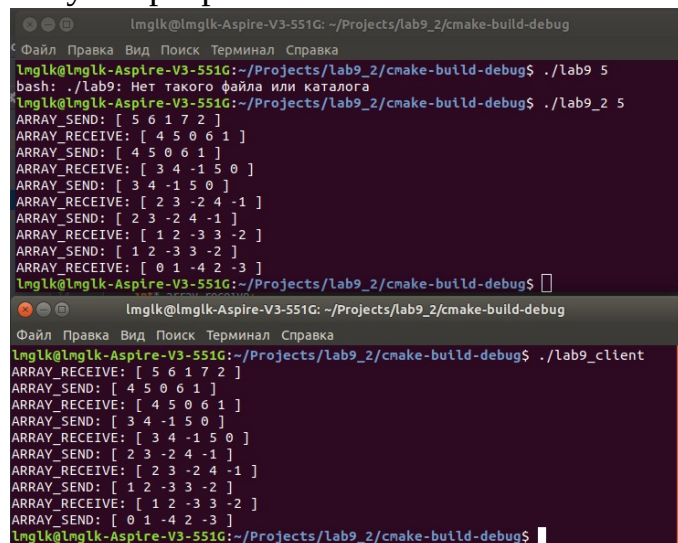
array_receive = (int*)shmat(shmid_another, nullptr, 0);
if (array_receive == (int*) (-1)) {
    cout << "Ошибка. Не удалось отобразить разделяемую память в адресное
пространство процесса" << endl;
    return 1;
}

while (array_receive[0] != -1) {
    while (array_receive[0] == 1) {
        cout << "ARRAY_RECEIVE: [ ";
        for (int i = 1; i < 6; i++) {
            cout << array_receive[i] << " ";
        }
        cout << "]" << endl;
        array_receive[0] = 0;
        sleep(1);

        cout << "ARRAY_SEND: [ ";
        for (int i = 1; i <= 5; i++) {
            array_send[i] = array_receive[i] - 1;
            cout << array_send[i] << " ";
        }
        cout << "]" << endl;
        array_send[0] = 1;
        sleep(1);
    }
}
return 0;
}

```


Запуск программы:



```
lmgik@lmgik-Aspire-V3-551G: ~/Projects/lab9_2/cmake-build-debug
Файл Правка Вид Поиск Терминал Справка
lmgik@lmgik-Aspire-V3-551G:~/Projects/lab9_2/cmake-build-debug$ ./lab9 5
bash: ./lab9: Нет такого файла или каталога
lmgik@lmgik-Aspire-V3-551G:~/Projects/lab9_2/cmake-build-debug$ ./lab9_2 5
ARRAY_SEND: [ 5 6 1 7 2 ]
ARRAY_RECEIVE: [ 4 5 0 6 1 ]
ARRAY_SEND: [ 4 5 0 6 1 ]
ARRAY_RECEIVE: [ 3 4 -1 5 0 ]
ARRAY_SEND: [ 3 4 -1 5 0 ]
ARRAY_RECEIVE: [ 2 3 -2 4 -1 ]
ARRAY_SEND: [ 2 3 -2 4 -1 ]
ARRAY_RECEIVE: [ 1 2 -3 3 -2 ]
ARRAY_SEND: [ 1 2 -3 3 -2 ]
ARRAY_RECEIVE: [ 0 1 -4 2 -3 ]
lmgik@lmgik-Aspire-V3-551G:~/Projects/lab9_2/cmake-build-debug$ 
lmgik@lmgik-Aspire-V3-551G:~/Projects/lab9_2/cmake-build-debug
Файл Правка Вид Поиск Терминал Справка
lmgik@lmgik-Aspire-V3-551G:~/Projects/lab9_2/cmake-build-debug$ ./lab9_client
ARRAY_RECEIVE: [ 5 6 1 7 2 ]
ARRAY_SEND: [ 4 5 0 6 1 ]
ARRAY_RECEIVE: [ 4 5 0 6 1 ]
ARRAY_SEND: [ 3 4 -1 5 0 ]
ARRAY_RECEIVE: [ 3 4 -1 5 0 ]
ARRAY_SEND: [ 2 3 -2 4 -1 ]
ARRAY_RECEIVE: [ 2 3 -2 4 -1 ]
ARRAY_SEND: [ 1 2 -3 3 -2 ]
ARRAY_RECEIVE: [ 1 2 -3 3 -2 ]
ARRAY_SEND: [ 0 1 -4 2 -3 ]
lmgik@lmgik-Aspire-V3-551G:~/Projects/lab9_2/cmake-build-debug$
```

Вывод

В ходе выполнения лабораторной работы был ознакомлен с организацией разделяемой памяти и системными функциями, обеспечивающими обмен данными между процессами.