

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

ОТЧЁТ

по лабораторной работе №10

по дисциплине «Организация процессов и программирования в среде Linux»

Тема: СИНХРОНИЗАЦИЯ ПРОЦЕССОВ С ПОМОЩЬЮ СЕМАФОРОВ

Студент гр. 9308

Преподаватель

Соболев М.С.

Разумовский Г.В.

Санкт-Петербург,

2022

Оглавление

1. Введение.....	3
1.1. Введение.....	3
1.2. Порядок выполнения работы.....	3
1.3. Содержание отчёта.....	4
2. Тексты программ.....	5
2.1. writer.cpp.....	5
2.2. reader.cpp.....	12
3. Скриншоты работы каждой программы.....	17
4. Вывод.....	23
5. Список использованных источников.....	24

1. Введение

1.1. Введение

Тема работы: Синхронизация процессов с помощью семафоров.

Цель работы: Знакомство с организацией семафоров, системными функциями, обеспечивающими управление семафорами, и их использованием для решения задач взаимного исключения и синхронизации.

1.2. Порядок выполнения работы

1. Написать две программы (Поставщик и Потребитель), которые работают с циклическим буфером ограниченного размера, расположенным в разделяемой памяти. Доступ к буферу и синхронизация работы Поставщика и Потребителя должны быть реализованы с помощью семафоров. Поставщик выделяет буфер и семафоры, читает по одному символу из файла и записывает его в буфер. Потребитель считывает по одному символу из буфера и выводит их на экран. Если буфер пустой, то Потребитель должен пассивно ждать, пока Поставщик не занесёт туда хотя бы один символ. Если буфер полностью заполнен, то Поставщик должен пассивно ждать, пока Потребитель не извлечёт из него по крайней мере один символ. Поставщик заканчивает свою работу, как только прочитает последний символ из файла и убедится, что Потребитель его прочитал. Потребитель заканчивает свою работу при отсутствии символов в буфере и завершении работы Поставщика.

2. Откомпилировать программы Поставщик и Потребитель. Запустить их на разных терминалах.

3. Написать две программы, экземпляры которых запускаются параллельно и с различной частотой обращаются к общему файлу. Каждый процесс из первой группы (Писатель) пополняет файл определённой строкой символов и выводит её на экран вместе с именем программы. Процессы второй

группы (Читатели) считывают весь файл и выводят его на экран. Писатели имеют приоритет перед Читателями. Пока один Писатель записывает строку в файл, другим Писателям и всем Читателям запрещено обращение к файлу. Читатели могут одновременно читать файл, если нет Писателей, готовых к записи в файл. Писатель заканчивает работу, после того как выполнит N-кратную запись строки в файл. Читатель заканчивает работу после прочтения текущего содержимого файла. Синхронизация процессов должна выполняться с помощью семафоров.

4. Откомпилировать программы Читатель и Писатель. Запустить на разных терминалах несколько Писателей и Читателей.

Выбранные задания: 3, 4.

1.3. Содержание отчёта

Отчёт по лабораторной работе должен содержать:

1. Цель и задания.
2. Тексты программ.
3. Скриншоты работы каждой программы.

2. Тексты программ

2.1. writer.cpp

```
/*
 * ./writer number_of_strings interval_time
 *
 * number_of_strings
 *   Number of loops (cycles), i.e. how many times this (WRITER) program (process) will write strings in the file.
Integer number in the range [0; +inf].
 * interval_time
 *   Interval time (as argument for "sleep()" function) for every loop (cycle), i.e. how many time we will wait after each
iteration. Integer number in the range [-1; +inf].
 *
 */

// https://www.unix.com/aix/74632-how-clean-unused-semaphore.html
// WARNING
// if you have undeleted old semaphore w/ old data,
// and you start a new program, but there is infinite waiting
// input in terminal: "ipcrm -s <semaphore id>"
// usually id starts from 1, so you can try it
// ID IS NOT KEY (it's int ptr in program, not the key)

#include <iostream>
#include <fstream>
#include <string>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <unistd.h>
#include <sys/shm.h>

using namespace std;

/*
struct sembuf
{
    short sem_num; // Semaphore number
    short sem_op; // Operation on semaphore
}
```

```

    short sem_flg; // Operation flags
};
*/

/*
* https://www.opennet.ru/docs/RUS/ipcbook/node34.html
* http://ccfit.nsu.ru/~deviv/courses/unix/unix/ngc9deb.html
* sem_num: semaphore number (index, id), for whom we setting the operations/flags
* sem_op > 0: add sem_op
* sem_op < 0: decrease by |sem_op|, if there will be number < 0 after that, it will return an error
* sem_op == 0: compare to 0, if != 0, there will be an error
* sem_flg == IPC_NOWAIT == 2048: if some operation couldn't be finished, there will be an error, nothing will be
changed
* sem_flg == SEM_UNDO == 4096: test operation regime, even if it's successful or not, it will be reseted (undo) to
values before
*/

/*
* http://ccfit.nsu.ru/~deviv/courses/unix/unix/ngc9deb.html
* int semop(int semid, struct sembuf *sops, unsigned nsops);
* nsops -- the number of structures in the array pointed to by sops, nsops must always be greater than or equal to 1;
* semid -- semaphore set identifier received from semget;
*/

typedef struct sembuf SemaphoreBuffer;

int main (int argc, char* argv[])
{
    // ----- INITIALIZING & PREPARING -----

    // checking if there is wrong/incopatible arguments
    if (argv[1] == nullptr)
    {
        cout << "Syntax error. No number of strings argument has found!\n";
        exit(-1);
    }
    if (atoi(argv[1]) < 0)
    {
        cout << "Syntax error. Number of strings to write file argument must be in range [0; +inf)\n";
    }
}

```

```

        exit(-1);
    }

    // checking if there is wrong/incopatible arguments
    if (argv[2] == nullptr)
    {
        cout << "Syntax error. No interval time (for \"sleep()\" function) argument has found!\n";
        exit(-1);
    }
    if (atoi(argv[2]) < -1)
    {
        cout << "Syntax error. Interval time (for \"sleep()\" function) argument must be in range [-1; +inf)!\n";
        exit(-1);
    }

    /*
    * SEMAPHORE CONVENTIONS (for all 3 semaphores):
    * 0 -- FILE semaphore
    * 1 -- ACTIVE WRITERS semaphore
    * 2 -- ACTIVE READERS semaphore
    * 3 -- ACTIVE PROCESSES semaphore
    */

    int number_of_strings = atoi(argv[1]); // number of loops (cycles)
    int interval_time = atoi(argv[2]); // interval time (as argument for "sleep()" function) for every loop (cycle)
    int i = 0;
    int semaphore_ptr; // pointer to the semaphore
    int key_semaphore = 190; // semaphore key
    string filename = "shared_file.txt"; // name of the file to write strings
    ofstream local_file;

    SemaphoreBuffer semaphore_file_decrease = {0, -1, 0}; // DEcreasing ACCESS FOR ONLY 1 WRITER to
    FILE semaphore (flags = 0)
    SemaphoreBuffer semaphore_file_increase = {0, 1, 0}; // INcreasing ACCESS FOR ONLY 1 WRITER to
    FILE semaphore (flags = 0)
    SemaphoreBuffer semaphore_writers_decrease = {1, -1, 0}; // DEcreasing ACTIVE WRITERS semaphore
    (flags = 0)
    SemaphoreBuffer semaphore_writers_increase = {1, 1, 0}; // INcreasing ACTIVE WRITERS semaphore (flags
    = 0)

```

```

SemaphoreBuffer semaphore_readers_compare = {2, 0, 0}; // comparing ACTIVE READERS semaphore with
0 (flags = 0)

SemaphoreBuffer semaphore_processes_decrease = {3, -1, 0}; // DEcreasing ACTIVE PROCESSES
semaphore (flags = 0)

SemaphoreBuffer semaphore_processes_increase = {3, 1, 0}; // INcreasing ACTIVE PROCESSES semaphore
(flags = 0)

cout << "----- WRITER PROCESS NUMBER " << getpid() << " -----\\n";
cout << "----- FILENAME TO WRITE IS " << filename << " -----\\n";
cout << "----- SEMAPHORE KEY IS " << (key_semaphore == IPC_PRIVATE ? "IPC_PRIVATE = " +
to_string(key_semaphore) : to_string(key_semaphore)) << " -----\\n";

// ----- CREATING/OPENING SEMAPHORES -----

// https://ru.manpages.org/semget/2
// int semget(key_t key, int nsems, int semflg);
semaphore_ptr = semget(key_semaphore, 4, IPC_CREAT | IPC_EXCL | 0666); // creating set of 4 semaphores
(file, active writers, active readers, active processes)

if (semaphore_ptr != -1)
{
    cout << "----- SEMAPHORE ID = " << semaphore_ptr << " HAS BEEN CREATED BY
PROCESS " << getpid() << " -----\\n";
    // when we create the semaphore, increasing it +1
    // this semaphore is only for writers: for tracking the possibility of writig in the file at the moment
    // when we enter the cycle (the loop), we decreasing it -1, so it will be =0 (because we had +1 ONLY
when created),
    // so the other ones (the other writers) can't make -1, when they reach the same point in the cycle,
    // because there is no IPC_NOWAIT flag, which returns error immediately, so other programs-writers
MUST wait untill it will be +1,
    // so they could make -1 (when =0, they couldn't do that, because there couldn't be <0 in semaphore)
    semop(semaphore_ptr, &semaphore_file_increase, 1);
}
else
{
    semaphore_ptr = semget(key_semaphore, 4, IPC_CREAT); // opening semaphore
    if (semaphore_ptr != -1)
    {

```



```

        cout << "----- SEMAPHORE ID = " << semaphore_ptr << " HAS BEEN OPENED BY
PROCESS " << getpid() << " -----\\n";
    }
    else
    {
        cout << "----- SEMAPHORE HAS NOT BEEN OPENED BY PROCESS " << getpid()
<< " -----\\n";

        exit(-1);
    }
}

// ----- INCREASING PROCESSES SEMAPHORE NUMBER -----

// increase number of working processes,
// because we started working w/ file, so other programs will know,
// how many programs are active (i.e. not finished working w/ file part)
semop(semaphore_ptr, &semaphore_processes_increase, 1);

cout << "----- NUMBER OF PROCESSES, CONNECTED TO R/W FILE IS " << semctl(semaphore_ptr,
3, GETVAL, 0) << " -----\\n\\n";

// ----- WRITING FILE -----

for (i = 0; i < number_of_strings; i++)
{
    semop(semaphore_ptr, &semaphore_writers_increase, 1); // +1 writer process, who wants to write into
the file

    semop(semaphore_ptr, &semaphore_readers_compare, 1); // writer waits until reader will finish the
reading

    cout << "----- W/ PROCESS №" << getpid() << " IS WAITING FOR THE FILE SEMAPHORE
-----\\n";

    // [see the semaphore creation part, i've made more detailed explanations]
    semop(semaphore_ptr, &semaphore_file_decrease, 1); // decrease, so other processes-WRITERs
waiting to get the access to the file

```

```

        cout << "----- OPEN FILE \"" << filename << "\" TO W/ BY PROCESS №" << getpid() << "
BEGIN -----\n";
        local_file.open(filename, ios::app);
        cout << "----- OPEN FILE \"" << filename << "\" TO W/ BY PROCESS №" << getpid() << "
END -----\n";

        cout << "----- WRITE STRING №" << i << " BY PROCESS №" << getpid() << " BEGIN
-----\n";
        local_file << "Written by process №" << getpid() << ". String №" << i << "\n";
        cout << "Written by process №" << getpid() << ". String №" << i << "\n";
        cout << "----- WRITE STRING №" << i << " BY PROCESS №" << getpid() << " END -----\n";

        cout << "----- CLOSE FILE \"" << filename << "\" TO W/ BY PROCESS №" << getpid() << "
BEGIN -----\n";
        local_file.close();
        cout << "----- CLOSE FILE \"" << filename << "\" TO W/ BY PROCESS №" << getpid() << "
END -----\n";

        cout << "----- FILE & ACTIVE W/S SEMAPHORE RELEASING BY W/ №" << getpid() << "
BEGIN -----\n";
        // [see the semaphore creation part, i've made more detailed explanations]
        semop(semaphore_ptr, &semaphore_file_increase, 1); // freeing the file holding by this process-
WRITER
        semop(semaphore_ptr, &semaphore_writers_decrease, 1); // decreasing number of active writers of
file
        cout << "----- FILE & ACTIVE W/S SEMAPHORE RELEASING BY W/ №" << getpid() << "
END -----\n\n";

        sleep(interval_time); // sleeping before next iteration by the time passed in the argument
    }

    // ----- DECREASING PROCESSES SEMAPHORE NUMBER -----

    // decrease number of working processes, because we finished shared file part
    // decreasing number in the semaphore number 4, so other programs will know,
    // how many programs are active (i.e. not finished working w/ file part)

```

```

semop(semaphore_ptr, &semaphore_processes_decrease, 1);

// ----- CLEANING & TERMINATING -----

// so the last process, who see 0 in semaphore,
// will delete semaphore and shared memory segment
if (semctl(semaphore_ptr, 3, GETVAL, 0) == 0) // last process will delete semaphore and will free the shared
memory segment
{
    semctl(semaphore_ptr, IPC_RMID, 0); // deleting semaphore
    cout << "----- SEMAPHORE HAS BEED DELETED -----\\n";
}

return 0;
}

```

2.2. reader.cpp

```
/*
 * ./reader
 *
 */

#include <iostream>
#include <fstream>
#include <string>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <unistd.h>
#include <sys/shm.h>

using namespace std;

typedef struct sembuf SemaphoreBuffer;

int main (int argc, char* argv[])
{
    // ----- INITIALIZING & PREPARING -----

    /*
     * SEMAPHORE CONVENTIONS (for all 3 semaphores):
     * 0 -- FILE semaphore
     * 1 -- ACTIVE WRITERS semaphore
     * 2 -- ACTIVE READERS semaphore
     * 3 -- ACTIVE PROCESSES semaphore
     */

    int semaphore_ptr; // pointer to the semaphore
    int key_semaphore = 190; // semaphore key
    char local_buffer[80]; // buffer to read the file
    string filename = "shared_file.txt"; // name of the file to read strings
    ifstream local_file;

    SemaphoreBuffer semaphore_file_increase = {0, 1, 0}; // INcreasing ACCESS FOR ONLY 1 WRITER to
    FILE semaphore (flags = 0)
```

```

SemaphoreBuffer semaphore_writers_compare = {1, 0, 0}; // comparing ACTIVE WRITERS semaphore with
0 (flags = 0)

SemaphoreBuffer semaphore_readers_decrease = {2, -1, 0}; // DEcreasing ACTIVE READERS semaphore
(flags = 0)

SemaphoreBuffer semaphore_readers_increase = {2, 1, 0}; // INcreasing ACTIVE READERS semaphore
(flags = 0)

SemaphoreBuffer semaphore_processes_decrease = {3, -1, 0}; // DEcreasing ACTIVE PROCESSES
semaphore (flags = 0)

SemaphoreBuffer semaphore_processes_increase = {3, 1, 0}; // INcreasing ACTIVE PROCESSES semaphore
(flags = 0)

cout << "----- READER PROCESS NUMBER " << getpid() << " -----\\n";
cout << "----- FILENAME TO READ IS " << filename << " -----\\n";
cout << "----- SEMAPHORE KEY IS " << (key_semaphore == IPC_PRIVATE ? "IPC_PRIVATE = " +
to_string(key_semaphore) : to_string(key_semaphore)) << " -----\\n";

// ----- CREATING/OPENING SEMAPHORES -----

// https://ru.manpages.org/semget/2
// int semget(key_t key, int nsems, int semflg);
semaphore_ptr = semget(key_semaphore, 4, IPC_CREAT | IPC_EXCL | 0666); // creating set of 4 semaphores
(file, active writers, active readers, active processes)

if (semaphore_ptr != -1)
{
    cout << "----- SEMAPHORE ID = " << semaphore_ptr << " HAS BEEN CREATED BY
PROCESS " << getpid() << " -----\\n";
    // when we create the semaphore, increasing it +1
    // this semaphore is only for writers: for tracking the possibility of writig in the file at the moment
    // when we enter the cycle (the loop), we decreasig it -1, so it will be =0 (because we had +1 ONLY
when created),
    // so the other ones (the other writers) can't make -1, when they reach the same point in the cycle,
    // because there is no IPC_NOWAIT flag, which returns error immediately, so other programs-writers
MUST wait untill it will be +1,
    // so they could make -1 (when =0, they couldn't do that, because there couldn't be <0 in semaphore)
    semop(semaphore_ptr, &semaphore_file_increase, 1);
}
else

```

```

{
    semaphore_ptr = semget(key_semaphore, 4, IPC_CREAT); // opening semaphore
    if (semaphore_ptr != -1)
    {
        cout << "----- SEMAPHORE ID = " << semaphore_ptr << " HAS BEEN OPENED BY
PROCESS " << getpid() << " -----\\n";
    }
    else
    {
        cout << "----- SEMAPHORE HAS NOT BEEN OPENED BY PROCESS " << getpid()
<< " -----\\n";
        exit(-1);
    }
}

// ----- INCREASING PROCESSES SEMAPHORE NUMBER -----

// increase number of working processes,
// because we started working w/ file, so other programs will know,
// how many programs are active (i.e. not finished working w/ file part)
semop(semaphore_ptr, &semaphore_processes_increase, 1);

cout << "----- NUMBER OF PROCESSES, CONNECTED TO R/W FILE IS " << semctl(semaphore_ptr,
3, GETVAL, 0) << " -----\\n\\n";

// ----- READING FILE -----

// here we are waiting only writers, but we aren't tracking other readers
// (as it was for writers w/ semaphore, where it increasing +1 in the creation part),
// because reading operation (instead of writing) doesn't require that
// so we have situation, where if NO writers -- ALL readers read,
// but if there IS writer -- ONLY 1 reader reads
// it's because in this situation there is a queue of processes, where (most probably) writer will be next, not
reader

cout << "----- R/ PROCESS №" << getpid() << " IS WAITING FOR THE READY W/ PROCS -----\\
n";

```

```
semop(semaphore_ptr, &semaphore_writers_compare, 1); // reader waits until writer will finish the reading
semop(semaphore_ptr, &semaphore_readers_increase, 1); // +1 writer process, who wants to read from the file
```

```
cout << "----- OPEN FILE \" << filename << "\" TO R/ BY PROCESS №" << getpid() << " BEGIN
-----\n";
local_file.open(filename);
cout << "----- OPEN FILE \" << filename << "\" TO R/ BY PROCESS №" << getpid() << " END
-----\n";
```

```
cout << "----- READ STRINGS BY PROCESS №" << getpid() << " BEGIN -----\n";
while (local_file.getline(local_buffer, 80))
{
    cout << local_buffer << "\n";
    sleep(1);
}
cout << "----- READ STRINGS BY PROCESS №" << getpid() << " END -----\n";
```

```
cout << "----- CLOSE FILE \" << filename << "\" TO R/ BY PROCESS №" << getpid() << " BEGIN
-----\n";
local_file.close();
cout << "----- CLOSE FILE \" << filename << "\" TO R/ BY PROCESS №" << getpid() << " END
-----\n\n";
```

```
semop(semaphore_ptr, &semaphore_readers_decrease, 1); // decreasing number of active readers of file
```

```
// ----- DECREASING PROCESSES SEMAPHORE NUMBER -----
```

```
// decrease number of working processes, because we finished shared file part
// decreasing number in the semaphore number 4, so other programs will know,
// how many programs are active (i.e. not finished working w/ file part)
semop(semaphore_ptr, &semaphore_processes_decrease, 1);
```

```
// ----- CLEANING & TERMINATING -----
```

```
// so the last process, who see 0 in semaphore,
```

```

// will delete semaphore and shared memory segment
if (semctl(semaphore_ptr, 3, GETVAL, 0) == 0) // last process will delete semaphore and will free the shared
memory segment
{
    semctl(semaphore_ptr, IPC_RMID, 0); // deleting semaphore
    cout << "----- SEMAPHORE HAS BEED DELETED -----\\n";
}

return 0;
}

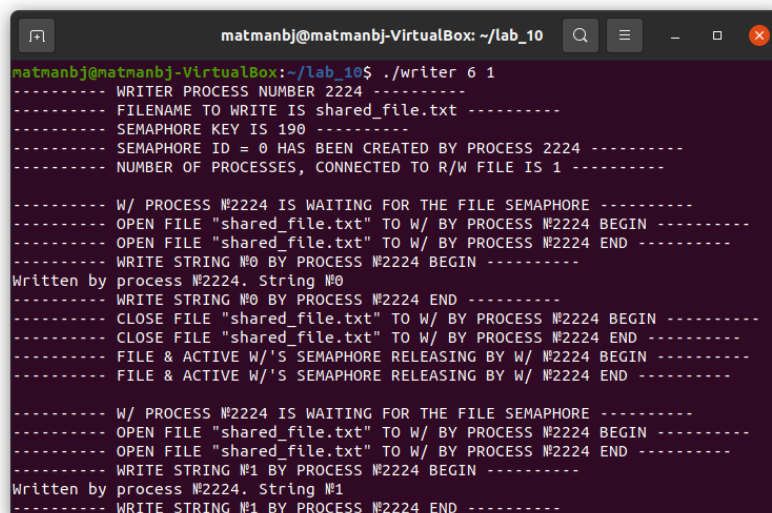
```


3. Скриншоты работы каждой программы

Запустим программы в следующем порядке: писатель, читатель, писатель, читатель. Все программы будут запускаться параллельно (последовательно в определённом порядке) в различных терминалах, не дожидаясь окончания работы остальных программ.

Программа-писатель «writer» запускается с параметрами количества строк, записываемых в файл, и времени ожидания (перед переходом на следующую итерацию) после завершения очередной итерации цикла. Программа-читатель «reader» запускается без параметров. Команды запуска программ в соответствующем порядке:

1. «./writer 6 1».
2. «./reader».
3. «./writer 7 1».
4. «./reader».



```
matmanbj@matmanbj-VirtualBox: ~/lab_10
matmanbj@matmanbj-VirtualBox:~/lab_10$ ./writer 6 1
----- WRITER PROCESS NUMBER 2224 -----
----- FILENAME TO WRITE IS shared_file.txt -----
----- SEMAPHORE KEY IS 190 -----
----- SEMAPHORE ID = 0 HAS BEEN CREATED BY PROCESS 2224 -----
----- NUMBER OF PROCESSES, CONNECTED TO R/W FILE IS 1 -----

----- W/ PROCESS №2224 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- WRITE STRING №0 BY PROCESS №2224 BEGIN -----
Written by process №2224. String №0
----- WRITE STRING №0 BY PROCESS №2224 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 END -----

----- W/ PROCESS №2224 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- WRITE STRING №1 BY PROCESS №2224 BEGIN -----
Written by process №2224. String №1
----- WRITE STRING №1 BY PROCESS №2224 END -----
```

Рисунок 1. Запуск программы-писателя с командой «./writer 6 1»

```
matmanbj@matmanbj-VirtualBox: ~/lab_10
----- WRITE STRING №1 BY PROCESS №2224 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 END -----

----- W/ PROCESS №2224 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- WRITE STRING №2 BY PROCESS №2224 BEGIN -----
Written by process №2224. String №2
----- WRITE STRING №2 BY PROCESS №2224 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 END -----

----- W/ PROCESS №2224 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- WRITE STRING №3 BY PROCESS №2224 BEGIN -----
Written by process №2224. String №3
----- WRITE STRING №3 BY PROCESS №2224 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
```

Рисунок 2. Запуск программы-писателя с командой «./writer 6 1»

```
matmanbj@matmanbj-VirtualBox: ~/lab_10
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 END -----

----- W/ PROCESS №2224 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- WRITE STRING №4 BY PROCESS №2224 BEGIN -----
Written by process №2224. String №4
----- WRITE STRING №4 BY PROCESS №2224 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 END -----

----- W/ PROCESS №2224 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- WRITE STRING №5 BY PROCESS №2224 BEGIN -----
Written by process №2224. String №5
----- WRITE STRING №5 BY PROCESS №2224 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
```

Рисунок 3. Запуск программы-писателя с командой «./writer 6 1»

```
matmanbj@matmanbj-VirtualBox: ~/lab_10

----- W/ PROCESS №2224 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- WRITE STRING №4 BY PROCESS №2224 BEGIN -----
Written by process №2224. String №4
----- WRITE STRING №4 BY PROCESS №2224 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 END -----

----- W/ PROCESS №2224 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- WRITE STRING №5 BY PROCESS №2224 BEGIN -----
Written by process №2224. String №5
----- WRITE STRING №5 BY PROCESS №2224 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2224 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2224 END -----

matmanbj@matmanbj-VirtualBox:~/lab_10$
```

Рисунок 4. Запуск программы-писателя с командой «./writer 6 1»

```
matmanbj@matmanbj-VirtualBox: ~/lab_10

matmanbj@matmanbj-VirtualBox:~/lab_10$ ./reader
----- READER PROCESS NUMBER 2226 -----
----- FILENAME TO READ IS shared_file.txt -----
----- SEMAPHORE KEY IS 190 -----
----- SEMAPHORE ID = 0 HAS BEEN OPENED BY PROCESS 2226 -----
----- NUMBER OF PROCESSES, CONNECTED TO R/W FILE IS 2 -----

----- R/ PROCESS №2226 IS WAITING FOR THE READY W/ PROCS -----
----- OPEN FILE "shared_file.txt" TO R/ BY PROCESS №2226 BEGIN -----
----- OPEN FILE "shared_file.txt" TO R/ BY PROCESS №2226 END -----
----- READ STRINGS BY PROCESS №2226 BEGIN -----
Written by process №2224. String №0
----- READ STRINGS BY PROCESS №2226 END -----
----- CLOSE FILE "shared_file.txt" TO R/ BY PROCESS №2226 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO R/ BY PROCESS №2226 END -----

matmanbj@matmanbj-VirtualBox:~/lab_10$
```

Рисунок 5. Запуск программы-читателя с командой «./reader»

```
matmanbj@matmanbj-VirtualBox: ~/lab_10
matmanbj@matmanbj-VirtualBox:~/lab_10$ ./writer 7 1
----- WRITER PROCESS NUMBER 2227 -----
----- FILENAME TO WRITE IS shared_file.txt -----
----- SEMAPHORE KEY IS 190 -----
----- SEMAPHORE ID = 0 HAS BEEN OPENED BY PROCESS 2227 -----
----- NUMBER OF PROCESSES, CONNECTED TO R/W FILE IS 3 -----

----- W/ PROCESS №2227 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- WRITE STRING №0 BY PROCESS №2227 BEGIN -----
Written by process №2227. String №0
----- WRITE STRING №0 BY PROCESS №2227 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 END -----

----- W/ PROCESS №2227 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- WRITE STRING №1 BY PROCESS №2227 BEGIN -----
Written by process №2227. String №1
----- WRITE STRING №1 BY PROCESS №2227 END -----
```

Рисунок 6. Запуск программы-писателя с командой «./writer 7 1»

```
matmanbj@matmanbj-VirtualBox: ~/lab_10
----- WRITE STRING №1 BY PROCESS №2227 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 END -----

----- W/ PROCESS №2227 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- WRITE STRING №2 BY PROCESS №2227 BEGIN -----
Written by process №2227. String №2
----- WRITE STRING №2 BY PROCESS №2227 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 END -----

----- W/ PROCESS №2227 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- WRITE STRING №3 BY PROCESS №2227 BEGIN -----
Written by process №2227. String №3
----- WRITE STRING №3 BY PROCESS №2227 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
```

Рисунок 7. Запуск программы-писателя с командой «./writer 7 1»

```
matmanbj@matmanbj-VirtualBox: ~/lab_10
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 END -----

----- W/ PROCESS №2227 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- WRITE STRING №4 BY PROCESS №2227 BEGIN -----
Written by process №2227. String №4
----- WRITE STRING №4 BY PROCESS №2227 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 END -----

----- W/ PROCESS №2227 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- WRITE STRING №5 BY PROCESS №2227 BEGIN -----
Written by process №2227. String №5
----- WRITE STRING №5 BY PROCESS №2227 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
```

Рисунок 8. Запуск программы-писателя с командой «./writer 7 1»

```
matmanbj@matmanbj-VirtualBox: ~/lab_10
----- W/ PROCESS №2227 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- WRITE STRING №5 BY PROCESS №2227 BEGIN -----
Written by process №2227. String №5
----- WRITE STRING №5 BY PROCESS №2227 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 END -----

----- W/ PROCESS №2227 IS WAITING FOR THE FILE SEMAPHORE -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- OPEN FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- WRITE STRING №6 BY PROCESS №2227 BEGIN -----
Written by process №2227. String №6
----- WRITE STRING №6 BY PROCESS №2227 END -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO W/ BY PROCESS №2227 END -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 BEGIN -----
----- FILE & ACTIVE W/'S SEMAPHORE RELEASING BY W/ №2227 END -----

----- SEMAPHORE HAS BEED DELETED -----
matmanbj@matmanbj-VirtualBox:~/lab_10$
```

Рисунок 9. Запуск программы-писателя с командой «./writer 7 1»

```
matmanbj@matmanbj-VirtualBox: ~/lab_10
matmanbj@matmanbj-VirtualBox:~/lab_10$ ./reader
----- READER PROCESS NUMBER 2228 -----
----- FILENAME TO READ IS shared_file.txt -----
----- SEMAPHORE KEY IS 190 -----
----- SEMAPHORE ID = 0 HAS BEEN OPENED BY PROCESS 2228 -----
----- NUMBER OF PROCESSES, CONNECTED TO R/W FILE IS 3 -----

----- R/ PROCESS №2228 IS WAITING FOR THE READY W/ PROCS -----
----- OPEN FILE "shared_file.txt" TO R/ BY PROCESS №2228 BEGIN -----
----- OPEN FILE "shared_file.txt" TO R/ BY PROCESS №2228 END -----
----- READ STRINGS BY PROCESS №2228 BEGIN -----
Written by process №2224. String №0
Written by process №2224. String №1
Written by process №2227. String №0
Written by process №2224. String №2
Written by process №2227. String №1
----- READ STRINGS BY PROCESS №2228 END -----
----- CLOSE FILE "shared_file.txt" TO R/ BY PROCESS №2228 BEGIN -----
----- CLOSE FILE "shared_file.txt" TO R/ BY PROCESS №2228 END -----

matmanbj@matmanbj-VirtualBox:~/lab_10$
```

Рисунок 10. Запуск программы-читателя с командой «./reader»

```
matmanbj@matmanbj-VirtualBox: ~/lab_10
matmanbj@matmanbj-VirtualBox:~/lab_10$ ./writer
Syntax error. No number of strings argument has found!
matmanbj@matmanbj-VirtualBox:~/lab_10$ ./writer 5
Syntax error. No interval time (for "sleep()" function) argument has found!
matmanbj@matmanbj-VirtualBox:~/lab_10$ ./writer -1
Syntax error. Number of strings to write file argument must be in range [0; +inf
)!
matmanbj@matmanbj-VirtualBox:~/lab_10$ ./writer 0 -2
Syntax error. Interval time (for "sleep()" function) argument must be in range [
-1; +inf)!
matmanbj@matmanbj-VirtualBox:~/lab_10$
```

Рисунок 11. Запуск программы-писателя с командой, содержащей неверные параметры

4. Вывод

В ходе выполнения лабораторной работы №10 «Синхронизация процессов с помощью семафоров» были изучены системные функции, отвечающие за создание («semget»), подключение («semget») и удаление («semctl») семафора, а также за его управление («semop»), изменение («semop») и чтение («semctl»). Был произведён запуск программ в определённом порядке, чтобы при этом они работали параллельно в различных терминалах, выводя записанные в один файл или считанные из одного файла строки в терминал. Также при запросе на запись от программы-писателя (программ-писателей) читать могла только одна программа-читатель, а если запросов на запись от программы-писателя (программ-писателей) не было, то читать файл могли все программы-читатели. Таким образом и было произведено знакомство с организацией семафоров, системными функциями, обеспечивающими управление семафорами, и их использованием для решения задач взаимного исключения и синхронизации.

5. Список использованных источников

1. Онлайн-курс «Организация процессов и программирование в среде Linux» в LMS Moodle [сайт]. URL: <https://vec.etu.ru/moodle/course/view.php?id=9703>.

2. Разумовский Г.В. Организация процессов и программирование в среде Linux: учебно-методическое пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2018. 40с.