

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

**ОТЧЕТ**  
**по лабораторной работе №10**  
**по дисциплине «Организация процессов и программирование**  
**в среде Linux»**  
**Тема: Синхронизация процессов с помощью семафоров**

Студент гр. 8308

\_\_\_\_\_

Петров Г.А.

Преподаватель

\_\_\_\_\_

Разумовский Г.В.

Санкт-Петербург

2021

## **Цель работы**

Целью лабораторной работы является знакомство с организацией семафоров, системными функциями, обеспечивающими управление семафорами, и их использованием для решения задач взаимного исключения и синхронизации.

## **Задание**

Напишите две программы, экземпляры которых запускаются параллельно и с разной частотой обращаются к общему файлу. Каждый процесс из первой группы (Писатель) пополняет файл определенной строкой символов и выводит ее на экран вместе с именем программы. Процессы второй группы (Читатели) считывают строки из файла и выводят их на экран при условии отсутствия ожидающих запись Писателей. Пока один Писатель записывает строку в файл, другим Писателям и всем Читателям запрещено обращение к файлу. Если Писатели не пишут в файл, то разрешается одновременная работа всех Читателей. Писатели должны ожидать, пока не закончат работу запущенные Читатели. Писатель заканчивает работу после того как выполнит N-кратную запись строки в файл. Работа Читателя завершается, когда он прочитал весь текущий файл. Синхронизация процессов должна выполняться с помощью семафоров.

## **Примеры выполнения программы**

Программы были разработаны и откомпилированы. После чего были запущены два писателя и четыре читателя через четыре терминала. Результаты работы программ приведены на рисунках 1-4. Также представлена распечатка файла TEXT.txt.

TEXT.txt

```
Писатель PID: 1457 Запись номер 0
Писатель PID: 1458 Запись номер 0
Писатель PID: 1457 Запись номер 1
Писатель PID: 1458 Запись номер 1
Писатель PID: 1457 Запись номер 2
Писатель PID: 1458 Запись номер 2
Писатель PID: 1457 Запись номер 3
Писатель PID: 1458 Запись номер 3
Писатель PID: 1457 Запись номер 4
Писатель PID: 1458 Запись номер 4
```

```
grigory@grigory-virtualbox:~/Desktop
/lab10$ ./lab10_w
Семафор открыт
Писатель PID: 1458 Запись номер 0
Писатель PID: 1458 Запись номер 1
Писатель PID: 1458 Запись номер 2
Писатель PID: 1458 Запись номер 3
Писатель PID: 1458 Запись номер 4
Семафор уничтожен!
```

Рисунок 1

```
grigory@grigory-virtualbox:~/Desktop
/lab10$ ./lab10_w
Семафор создан
Семафор инициализирован
Семафор открыт
Писатель PID: 1457 Запись номер 0
Писатель PID: 1457 Запись номер 1
Писатель PID: 1457 Запись номер 2
Писатель PID: 1457 Запись номер 3
Писатель PID: 1457 Запись номер 4
```

Рисунок 2

```
grigory@grigory-virtualbox:~/Desktop
/lab10$ ./lab10_r
Семафор открыт
-----
Писатель PID: 1457 Запись номер 0
Писатель PID: 1458 Запись номер 0
-----
grigory@grigory-virtualbox:~/Desktop
/lab10$ ./lab10_r
Семафор создан
Семафор инициализирован
Семафор открыт
-----
Писатель PID: 1457 Запись номер 0
Писатель PID: 1458 Запись номер 0
Писатель PID: 1457 Запись номер 1
Писатель PID: 1458 Запись номер 1
Писатель PID: 1457 Запись номер 2
Писатель PID: 1458 Запись номер 2
Писатель PID: 1457 Запись номер 3
Писатель PID: 1458 Запись номер 3
Писатель PID: 1457 Запись номер 4
Писатель PID: 1458 Запись номер 4
-----
```

Рисунок 3

```
grigory@grigory-virtualbox:~/Desktop
/lab10$ ./lab10_r
Семафор открыт
-----
Писатель PID: 1457 Запись номер 0
Писатель PID: 1458 Запись номер 0
Писатель PID: 1457 Запись номер 1
Писатель PID: 1458 Запись номер 1
-----
grigory@grigory-virtualbox:~/Desktop
/lab10$ ./lab10_r
Семафор открыт
-----
Писатель PID: 1457 Запись номер 0
Писатель PID: 1458 Запись номер 0
Писатель PID: 1457 Запись номер 1
Писатель PID: 1458 Запись номер 1
Писатель PID: 1457 Запись номер 2
Писатель PID: 1458 Запись номер 2
Писатель PID: 1457 Запись номер 3
Писатель PID: 1458 Запись номер 3
-----
```

Рисунок 4

# Исходный код программ

## lab10\_w.cpp

```
#include "fun.h"
#include <unistd.h>
#include <fstream>

int main()
{
    struct sembuf sb;
    getSemafor();

    for(int i=0;i<WRITE_COUNT;++i)
    {
        sleep(1);

        //Увеличение количества активных писателей
        sb.sem_num=1; sb.sem_flg=0; sb.sem_op=1;
        if(semop(SID,&sb,1)<0)
        {
            perror("Error in function semop(add active writer)");
            exit(2);
        }
        sb.sem_num=2; sb.sem_flg=0; sb.sem_op=0;
        semop(SID,&sb,1); //ожидание окончания работы активных читателей
        sb.sem_num=0; sb.sem_flg=0; sb.sem_op=-1;
        if(semop(SID,&sb,1)<0) //блокировка ресурса на запись
        {
            perror("Error in function semop(block writer)");
            exit(2);
        }

        std::ofstream fout(FNAME,std::ios_base::app);
        fout<<"Писатель PID: "<<getpid()<<" Запись номер "<<i<<std::endl;
        std::cout<<"Писатель PID: "<<getpid()<<" Запись номер "<<i<<std::endl;
        fout.close();

        sb.sem_num=0; sb.sem_flg=0; sb.sem_op=1;
        if(semop(SID,&sb,1)<0) //разблокировка ресурса на запись
        {
            perror("Error in function semop(unblock writer)");
            exit(2);
        }
        sb.sem_num=1; sb.sem_flg=0; sb.sem_op=-1;
        if(semop(SID,&sb,1)<0) //отпустить флаг активного писателя
        {
            perror("Error in function semop(add active writer)");
            exit(2);
        }
    }

    destructSemafor();
    return 0;
}
```

## lab10\_r.cpp

```
#include "fun.h"
#include <unistd.h>
#include <fstream>

int main()
{
    struct sembuf sb;
```

```

getSemafor();

std::string str;

//Начало работы только если нет активного писателя или
//писателей ожидающих окончания активных читателей
sb.sem_num=1; sb.sem_flg=0; sb.sem_op=0;
semop(SID,&sb,1);
sb.sem_num=2; sb.sem_flg=0; sb.sem_op=1;
if(semop(SID,&sb,1)<0)//добавление активного читателя
{
    perror("Error in function semop(add activ reader)");
    exit(2);
}

sleep(1);
std::ifstream fin(FNAME);
std::cout<<"-----"<<std::endl;
while(getline(fin,str))
    std::cout<<str<<std::endl;
std::cout<<"-----"<<std::endl;
fin.close();

sb.sem_num=2; sb.sem_flg=0; sb.sem_op=-1;
if(semop(SID,&sb,1)<0)//уменьшение числа активных читателей
{
    perror("Error in function semop(del activ reader)");
    exit(2);
}

destructSemafor();
return 0;
}

```

## fun.cpp

```

#include <iostream>
#include <sys/sem.h>
#include <algorithm>
#include <semaphore.h>
#define FNAME "TEXT.txt"
#define KEY 999
#define WRITE_COUNT 5

int SID=-1;

void getSemafor();//открыть\создать семафор
void destructSemafor();//уничтожение семафора

void getSemafor()
{
    SID=semget(KEY,4,0666);
    if(SID<0)
    {
        SID=semget(KEY,4,0666|IPC_CREAT); 6
    }
}

```

```

    if(SID<0)
    {
        perror("error in function [semget()]");
        exit(1);
    }
    std::cout<<"Семафор создан"<<std::endl;
    //первый семафор в 1 - ресурс свободен на запись, в 0 - ресурс занят
    //второй семафор - счетчик активных писателей
    //третий семафор - счетчик активных читателей
    //четвертый семафор - счетчик процессов работающих с множ-ным семафором
    short val[4]={1,0,0,0};
    semctl(SID,4,SETALL,val);
    std::cout<<"Семафор инициализирован"<<std::endl;
}
std::cout<<"Семафор открыт"<<std::endl;

struct sembuf sb;//увеличить количество работающих процессов
sb.sem_num=3; sb.sem_flg=0; sb.sem_op=1;
if(semop(SID,&sb,1)<0)
{
    perror("Error in function semop(add activ proc)");
    exit(2);
}
}

void destructSemafor()//уничтожение семафора
{
    struct sembuf sb;
    sb.sem_num=3; sb.sem_flg=0; sb.sem_op=-1;
    if(semop(SID,&sb,1)<0)//минус 1 работающий процесс
    {
        perror("Error in function semop(minus activ proc)");
        exit(2);
    }
    if(semctl(SID,3,GETVAL)==0)//это последний процесс
    {
        semctl(SID,IPC_RMID,0);//уничтожение множ-го семафора
        std::cout<<"Семафор уничтожен!"<<std::endl;
    }
}
}

```

## **Вывод**

При выполнении лабораторной работы для решения задач взаимного исключения и синхронизации изучены и использованы семафоры и системные функции, обеспечивающие управление семафорами. Программа разработанная в соответствии с заданием, работает корректно.