

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. 9308

Яловега Н.В.

Преподаватель

Разумовский Г.В.

Санкт-Петербург

2022

Цель работы

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

Задание

1. Написать две программы, экземпляры которых запускаются параллельно и с различной 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 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.

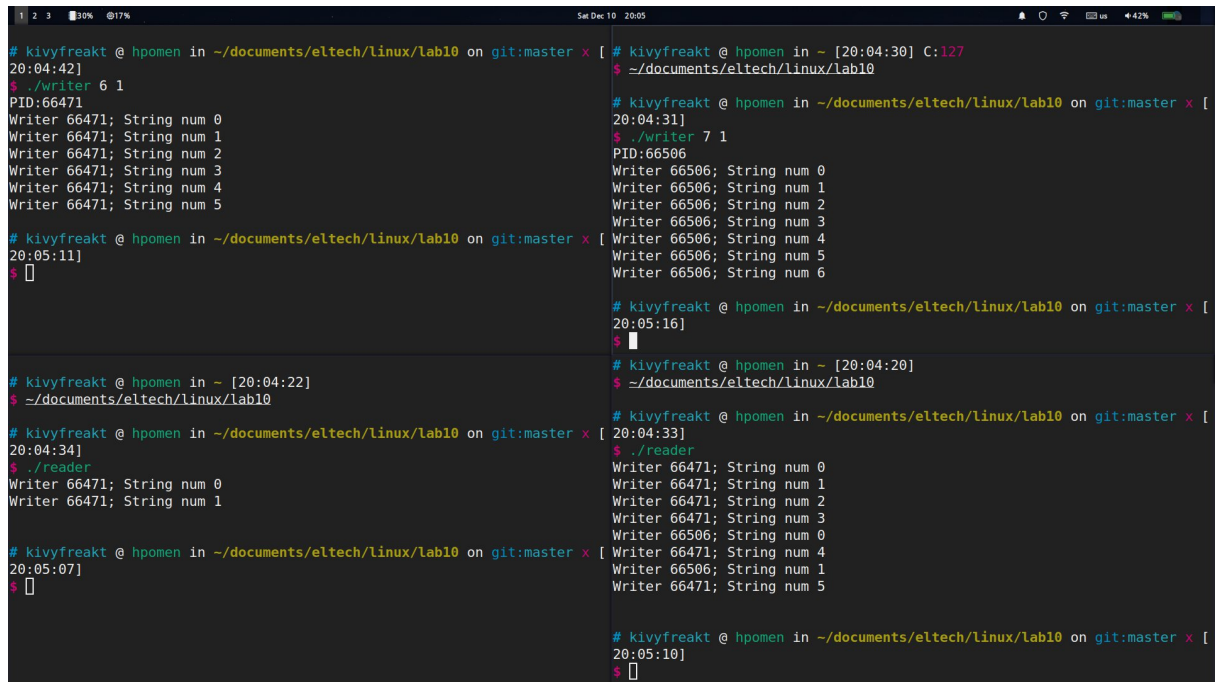
2. Откомпилировать программы Читатель и Писатель. Запустить на разных терминалах несколько Писателей и Читателей.

Порядок выполнения работы

Программы синхронизируются с помощью множественного семафора. Семафор 0 представляет собой мьютекс, который используется для ограничения записи в файл. Семафор 1 служит для индикации окончания работы всех процессов, чтобы последний процесс выполнил `IPC_RMID`. Семафоры 3 и 4 показывают количество работающих писателей и читателей соответственно. Таким образом, процесс писатель прежде чем начать запись должен проверить, семафор читателей 0, а процесс читатель перед чтением должен проверять семафор писателей на 0.

Примеры выполнения программы

Скриншоты выполнения программ представлены на рис 1.



```
# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab10 on git:master x [20:04:42]
$ ./writer 6 1
PID:66471
Writer 66471; String num 0
Writer 66471; String num 1
Writer 66471; String num 2
Writer 66471; String num 3
Writer 66471; String num 4
Writer 66471; String num 5

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab10 on git:master x [20:05:11]
$

# kivyfreakt @ hpomen in [20:04:22]
$ ~/documents/eltech/linux/lab10

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab10 on git:master x [20:04:34]
$ ./reader
Writer 66471; String num 0
Writer 66471; String num 1

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab10 on git:master x [20:05:07]
$

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab10 on git:master x [20:04:31]
$ ./writer 7 1
PID:66506
Writer 66506; String num 0
Writer 66506; String num 1
Writer 66506; String num 2
Writer 66506; String num 3
Writer 66506; String num 4
Writer 66506; String num 5
Writer 66506; String num 6

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab10 on git:master x [20:05:16]
$

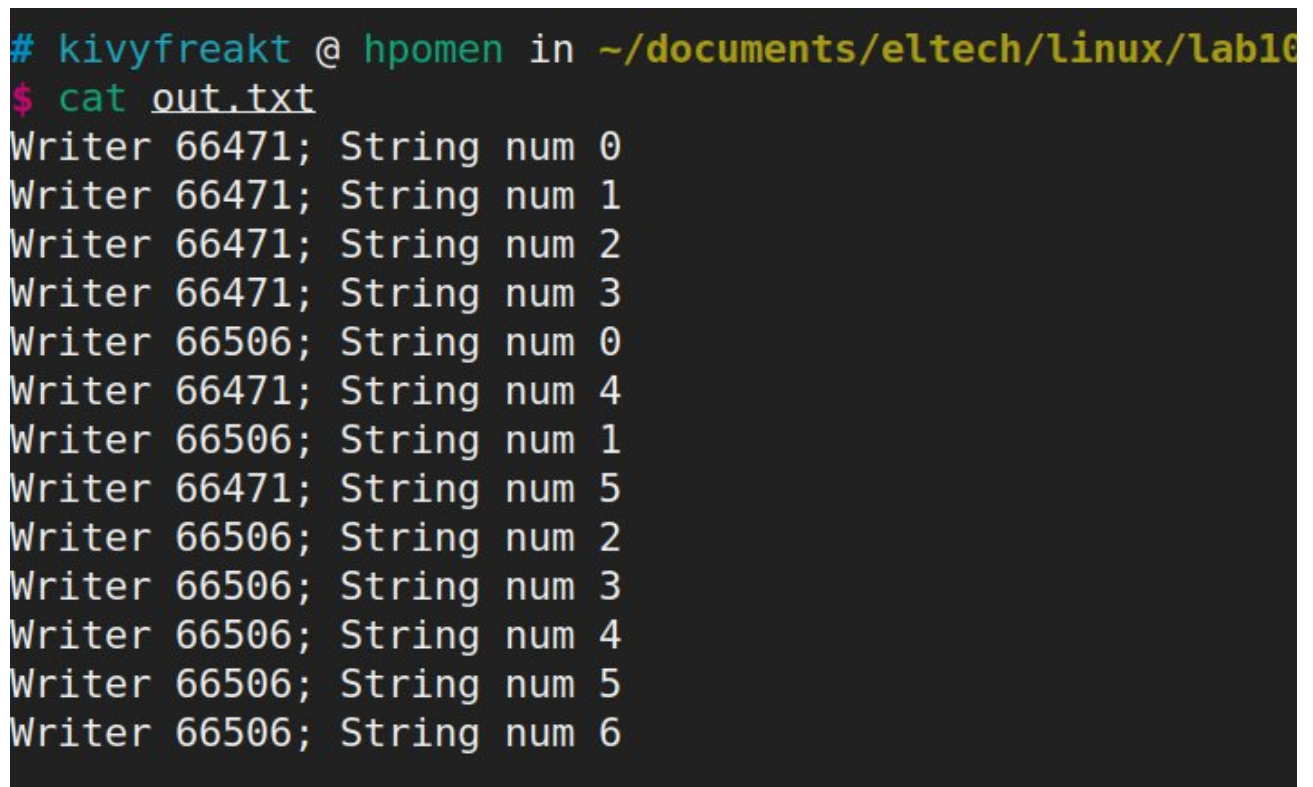
# kivyfreakt @ hpomen in [20:04:20]
$ ~/documents/eltech/linux/lab10

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab10 on git:master x [20:04:33]
$ ./reader
Writer 66471; String num 0
Writer 66471; String num 1
Writer 66471; String num 2
Writer 66471; String num 3
Writer 66506; String num 0
Writer 66471; String num 4
Writer 66506; String num 1
Writer 66471; String num 5

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab10 on git:master x [20:05:10]
$
```

Рисунок 1

Данные выходного файла представлены на рис. 4.



```
# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab10
$ cat out.txt
Writer 66471; String num 0
Writer 66471; String num 1
Writer 66471; String num 2
Writer 66471; String num 3
Writer 66506; String num 0
Writer 66471; String num 4
Writer 66506; String num 1
Writer 66471; String num 5
Writer 66506; String num 2
Writer 66506; String num 3
Writer 66506; String num 4
Writer 66506; String num 5
Writer 66506; String num 6
```

Рисунок 2

Исходный код

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

int main(int argc, char **argv)
{
    // 0 - индекс
    // 1 - операция (положительное + , отрицательное число - или нуль (получение))
    // 2 - флаги
    struct sembuf operation;

    int number = atoi(argv[1]);
    int sleep = atoi(argv[2]);

    // 5051 - key
    // 4 - количество семафоров (0 - занятость файла, 1 - программы, 2 - писатели, 3 - читатели)
    // IPC_CREAT | IPC_EXCL | 0666 - флаги
    int semaphore = semget(5051, 4, IPC_CREAT | IPC_EXCL | 0666);

    if (semaphore != -1)
    {
        operation = {0, 1, 0}; // файл доступен

        // semaphore – идентификатор семафора;
        // operation – массив операций;
        // 1 – число операций
        semop(semaphore, &operation, 1);
    }
    else
        semaphore = semget(5051, 4, IPC_CREAT | 0666);

    std::ofstream file("out.txt", std::ios::app);

    // программы увеличиваем
    operation = {1, 1, 0};
    semop(semaphore, &operation, 1);
```

```

int pid = getpid();
std::cout << "PID:" << pid << std::endl;

for (int i = 0; i < number; i++)
{
    // увеличиваем писателей
    operation = {2, 1, 0};
    semop(semaphore, &operation, 1);

    // проверка читателей
    operation = {3, 0, 0};
    semop(semaphore, &operation, 1);

    // занимаем файл
    operation = {0, -1, 0};
    semop(semaphore, &operation, 1);

    // пишем в файл
    file << "Writer " << pid << "; String num " << i << std::endl;
    std::cout << "Writer " << pid << "; String num " << i << std::endl;

    // открываем файл
    operation = {0, 1, 0};
    semop(semaphore, &operation, 1);

    // уменьшаем писателей
    operation = {2, -1, 0};
    semop(semaphore, &operation, 1);

    // ждем....
    sleep(slept);
}

// уменьшаем программы
operation = {1, -1, 0};
semop(semaphore, &operation, 1);

```

```
if (semctl(semaphore, 1, GETVAL, 0) == 0)
{
    semctl(semaphore, IPC_RMID, 0);
}

file.close();
return 0;
}
```

```

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

int main(int argc, char **argv)
{
    // 0 - индекс
    // 1 - операция (положительное + , отрицательное число - или нуль (получение))
    // 2 - флаги
    struct sembuf operation;

    int number = atoi(argv[1]);
    int sleep = atoi(argv[2]);

    // 5051 - key
    // 4 - количество семафоров (0 - занятость файла, 1 - программы, 2 - писатели, 3 - читатели)
    // IPC_CREAT | IPC_EXCL | 0666 - флаги
    int semaphore = semget(5051, 4, IPC_CREAT | IPC_EXCL | 0666);

    if (semaphore != -1)
    {
        operation = {0, 1, 0}; // файл доступен

        // semaphore – идентификатор семафора;
        // operation – массив операций;
        // 1 – число операций
        semop(semaphore, &operation, 1);
    }
    else
        semaphore = semget(5051, 4, IPC_CREAT | 0666);

    std::ofstream file("out.txt", std::ios::app);

    // программы увеличиваем
    operation = {1, 1, 0};
    semop(semaphore, &operation, 1);

```

```

int pid = getpid();
std::cout << "PID:" << pid << std::endl;

for (int i = 0; i < number; i++)
{
    // увеличиваем писателей
    operation = {2, 1, 0};
    semop(semaphore, &operation, 1);

    // проверка читателей
    operation = {3, 0, 0};
    semop(semaphore, &operation, 1);

    // занимаем файл
    operation = {0, -1, 0};
    semop(semaphore, &operation, 1);

    // пишем в файл
    file << "Writer " << pid << "; String num " << i << std::endl;
    std::cout << "Writer " << pid << "; String num " << i << std::endl;

    // открываем файл
    operation = {0, 1, 0};
    semop(semaphore, &operation, 1);

    // уменьшаем писателей
    operation = {2, -1, 0};
    semop(semaphore, &operation, 1);

    // ждем....
    sleep(sleept);
}

// уменьшаем программы
operation = {1, -1, 0};
semop(semaphore, &operation, 1);

if (semctl(semaphore, 1, GETVAL, 0) == 0)
{

```



```
    semctl(semaphore, IPC_RMID, 0);  
}  
  
file.close();  
return 0;  
}
```

Вывод

В ходе работы были изучены механизмы организации семафоров, системными функциями, обеспечивающими управление семафорами, и их использованием для решения задач взаимного исключения и синхронизации в операционной системе linux.