

**«Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
СПбГЭТУ «ЛЭТИ»**

Кафедра вычислительной техники

Отчет по по лабораторной работе №10 по дисциплине
«Организация процессов и программирование в среде Linux»
Тема: «СИНХРОНИЗАЦИЯ ПРОЦЕССОВ С ПОМОЩЬЮ
СЕМАФОРОВ»

Студент гр.8306

Преподаватель

Слепов А.Э.

Разумовский Г.В.

Санкт-Петербург

2021

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.

Результаты запуска программ для 2 писателей (20 и 10 строк) и 2 читателей приведены на рисунке 1.

```
artem@legion: ~/Documents/linux-labs/lab10/new
Процесс-писатель ожидает освобождения writer mutex
Процесс 33489 записал 16/19
Процесс-писатель освободил writer mutex

Процесс-писатель ожидает освобождения writer mutex
Процесс 33489 записал 17/19
Процесс-писатель освободил writer mutex

Процесс-писатель ожидает освобождения writer mutex
Процесс 33489 записал 18/19
Процесс-писатель освободил writer mutex

Процесс-писатель ожидает освобождения writer mutex
Процесс 33489 записал 19/19
Процесс-писатель освободил writer mutex

Семафор уничтожены
artem@legion: ~/Documents/linux-labs/lab10/new$

artem@legion: ~/Documents/linux-labs/lab10/new
Процесс-писатель освободил writer mutex

Процесс-писатель ожидает освобождения writer mutex
Процесс 33491 записал 6/9
Процесс-писатель освободил writer mutex

Процесс-писатель ожидает освобождения writer mutex
Процесс 33491 записал 7/9
Процесс-писатель освободил writer mutex

Процесс-писатель ожидает освобождения writer mutex
Процесс 33491 записал 8/9
Процесс-писатель освободил writer mutex

Процесс-писатель ожидает освобождения writer mutex
Процесс 33491 записал 9/9
Процесс-писатель освободил writer mutex

artem@legion: ~/Documents/linux-labs/lab10/new$

artem@legion: ~/Documents/linux-labs/lab10/new
Процесс-читатель 33492 прочитал строку №4
Процесс-писатель 33489 4/19

Процесс-читатель 33492 прочитал строку №5
Процесс-писатель 33491 0/9

Процесс-читатель 33492 прочитал строку №6
Процесс-писатель 33489 5/19

Процесс-читатель 33492 прочитал строку №7
Процесс-писатель 33491 1/9

Процесс-читатель 33492 прочитал строку №8
Процесс-писатель 33489 6/19

Процесс-читатель 33492 прочитал строку №9
Процесс-писатель 33491 2/9

artem@legion: ~/Documents/linux-labs/lab10/new$

artem@legion: ~/Documents/linux-labs/lab10/new
Процесс-читатель 33493 прочитал строку №11
Процесс-писатель 33491 3/9

Процесс-читатель 33493 прочитал строку №12
Процесс-писатель 33489 8/19

Процесс-читатель 33493 прочитал строку №13
Процесс-писатель 33491 4/9

Процесс-читатель 33493 прочитал строку №14
Процесс-писатель 33489 9/19

Процесс-читатель 33493 прочитал строку №15
Процесс-писатель 33491 5/9

artem@legion: ~/Documents/linux-labs/lab10/new$
```

Рисунок 1. Результаты запуска программы

Текстовый файл, в который писали писатели представлен на рисунке 2.

Открыть	output.txt
	~/Documents/linux-labs/L...
1	Процесс-писатель 33489 0/19
2	Процесс-писатель 33489 1/19
3	Процесс-писатель 33489 2/19
4	Процесс-писатель 33489 3/19
5	Процесс-писатель 33489 4/19
6	Процесс-писатель 33491 0/9
7	Процесс-писатель 33489 5/19
8	Процесс-писатель 33491 1/9
9	Процесс-писатель 33489 6/19
10	Процесс-писатель 33491 2/9
11	Процесс-писатель 33489 7/19
12	Процесс-писатель 33491 3/9
13	Процесс-писатель 33489 8/19
14	Процесс-писатель 33491 4/9
15	Процесс-писатель 33489 9/19
16	Процесс-писатель 33491 5/9
17	Процесс-писатель 33489 10/19
18	Процесс-писатель 33491 6/9
19	Процесс-писатель 33489 11/19
20	Процесс-писатель 33491 7/9
21	Процесс-писатель 33489 12/19
22	Процесс-писатель 33491 8/9
23	Процесс-писатель 33489 13/19
24	Процесс-писатель 33491 9/9
25	Процесс-писатель 33489 14/19
26	Процесс-писатель 33489 15/19
27	Процесс-писатель 33489 16/19
28	Процесс-писатель 33489 17/19
29	Процесс-писатель 33489 18/19
30	Процесс-писатель 33489 19/19

Рисунок 2. Выходной текстовый файл

Выводы

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 Ubuntu.

ПРИЛОЖЕНИЕ А

Текст программы писателя

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/types.h>
#include <sys/shm.h>

#define SEM_KEY 111

//0 - мьютекс файла
//1 - семафор numProcess
//2 - семафор numWriters
//3 - семафор numReaders
int semId;

//операции над семафорами
struct sembuf writer_dec_sem = {0, -1, 0};
struct sembuf writer_inc_sem = {0, 1, 0};

struct sembuf numwriters_dec = {2, -1, 0};
struct sembuf numwriters_inc = {2, 1, 0};
struct sembuf numreaders_dec = {3, -1, 0};
struct sembuf numreaders_inc = {3, 1, 0};

struct sembuf is_reader_zero = {3, 0, 0};

FILE *file;
int main(int argc, char **argv){
    if(argc == 2){
        int writeNum = atoi(argv[1]);

        int pid = getpid();
        struct sembuf sem_init = {0, 1, 0};
        semId = semget(SEM_KEY,4,IPC_CREAT|IPC_EXCL|0666);
        if(semId > 0){
            file = fopen("output.txt","w");
            fclose(file); //очистка файла
            printf("Процесс %d создал семафор\n",pid);
            semop(semId,&sem_init,1); //мьютекс файла
        }else{
            semId = semget(SEM_KEY,4,IPC_CREAT);
            printf("Используем существующий семафор %d\n", semId);
        }
        sem_init.sem_num = 1;
        semop(semId,&sem_init,1); //общее количество процессов

        //sem_init.sem_num = 2;
        //semop(semId,&sem_init,1); //кол-во писателей

        printf("semid: %d\n", semId);

        for(int i=0;i<writeNum;i++){
            semop(semId,&numwriters_inc, 1);

            semop(semId, &is_reader_zero, 1);
```

```

        //while(semctl(semId, 3, GETVAL, 0 ) > 0);

        printf("Процесс-писатель ожидает освобождения writer mutex\
n");

        semop(semId, &writer_dec_sem, 1);
        printf("\tПроцесс %d записал %d/%d\n", pid, i, writeNum-1);
        file = fopen("output.txt", "a");
        fprintf(file,"Процесс-писатель %d  %d/%d\n",pid, i, writeNum-
1);

        fclose(file);
        printf("Процесс-писатель освободил writer mutex\n\n");
        semop(semId, &writer_inc_sem, 1);

        semop(semId,&numwriters_dec, 1);
        sleep(1);
    }

    struct sembuf inc_all = {1, -1, 0};
    semop(semId, &inc_all, 1);
    if(semctl( semId, 1, GETVAL, 0 ) == 0){
        semctl(semId,IPC_RMID,0);

        printf("Семафор уничтожены\n");
    }
    else{
        printf("Недостаточное количество аргументов: ./name <количество опе-
раций>");
    }
    return 0;
}

```

ПРИЛОЖЕНИЕ Б

Текст программы читателя

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/types.h>
#include <sys/shm.h>

#define SEM_KEY 111

//0 - мьютекс файла
//1 - семафор numProcess
//2 - семафор numWriters
//3 - семафор numReaders
int semId;

//операции над семафорами
struct sembuf writer_dec_sem = {0, -1, 0};
struct sembuf writer_inc_sem = {0, 1, 0};

struct sembuf numwriters_dec = {2, -1, 0};
struct sembuf numwriters_inc = {2, 1, 0};
struct sembuf numreaders_dec = {3, -1, 0};
struct sembuf numreaders_inc = {3, 1, 0};

struct sembuf is_writer_zero = {2, 0, 0};

FILE *file;
int main(int argc, char **argv){
    int pid = getpid();
    semId = semget(SEM_KEY, 4, IPC_CREAT | IPC_EXCL | 0666);

    if(semId > 0){
        file = fopen("output.txt", "w");
        fclose(file); //очистка файла
        printf("Процесс %d создал семафор\n", pid);
    }else{
        semId = semget(SEM_KEY, 4, IPC_CREAT);
        printf("Используем существующий семафор %d\n", semId);
    }
    struct sembuf sem_init = {0, 1, 0};
    sem_init.sem_num = 1;
    semop(semId, &sem_init, 1); //общее количество процессов

    char flag = 1;
    int curStr = 0;
    char buf[256];

    semop(semId, &numreaders_inc, 1);
    printf("\nПроцесс читатель ждет, пока писатель запишет\n");

    semop(semId, &is_writer_zero, 1);
    //while(semctl( semId, 2, GETVAL, 0 ) > 0);

    file = fopen("output.txt", "r");
    int i = 0;
    i = 0;
```

```

while(fgets(buf, 256, file)){
    printf("Процесс-читатель %d прочитал строку №%d \n\t %s\n",pid, i,
buf);
    i+=1;
}
flag = 0;
fclose(file);
semop(semId, &numreaders_dec, 1);

struct sembuf inc_all = {1, -1, 0};
semop(semId, &inc_all, 1);
if(semctl( semId, 1, GETVAL, 0 ) == 0){
    semctl(semId,IPC_RMID,0);

    printf("Семафор уничтожены\n");
}
return 0;
}

```