Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

ОТЧЕТ
По лабораторной работе №7
Дисциплина «ОСиСП» за IV семестр

Выполнила:

студент группы ПО-3

Ковалёва А. И.

Проверила:

Давидюк Ю. И.

Брест 2020

Лабораторная работа 7
Семафоры

Вариант 12

Цель:

Ознакомиться с реализацией семафоров в ОС Linux.

Задание:

Написать две (или более) программы, которые, работая параллельно зациклено, обмениваются информацией согласно варианту. Передачу и получение информации каждым из процессов сопровождать выводом на экран нформации типа "процесс такой-то передал/получил такую-то информацию". Синхронизацию работы процессов реализовать с помощью семафоров. Учтите, что при организации совместного доступа к разделяемому ресурсу (например, файлу) вам понадобится применять, например, мьютексы.

Вариант 2, 12. Первый процесс в цикле ожидает ввода символа с потока stdin, после чего пишет в файл соответствующий символ, каждый раз открывая и закрывая за собой файл. Второй процесс забирает из файла символ и выводит его на экран несколько раз подряд.

Код программы:

main1.c

```
#include<stdio.h>
#include<fcntl.h>
#include<sys/types.h>
#include<unistd.h>
#include<stdlib.h>
#include<semaphore.h>
#include <pthread.h>

static pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

int main() {
    sem_t *sem1, *sem2;
    int f;

    if((sem1 = sem_open("semaphore1", O_RDWR|O_CREAT, 0666,1)) ==
SEM_FAILED) {
        printf("ERROR sem_open");
        return 1;
    }

    if((sem2 = sem_open("semaphore2", O_RDWR|O_CREAT, 0666,0)) ==
SEM_FAILED) {
        printf("ERROR sem_open");
        return 1;
    }
    // sem_unlink("semaphore1");
```

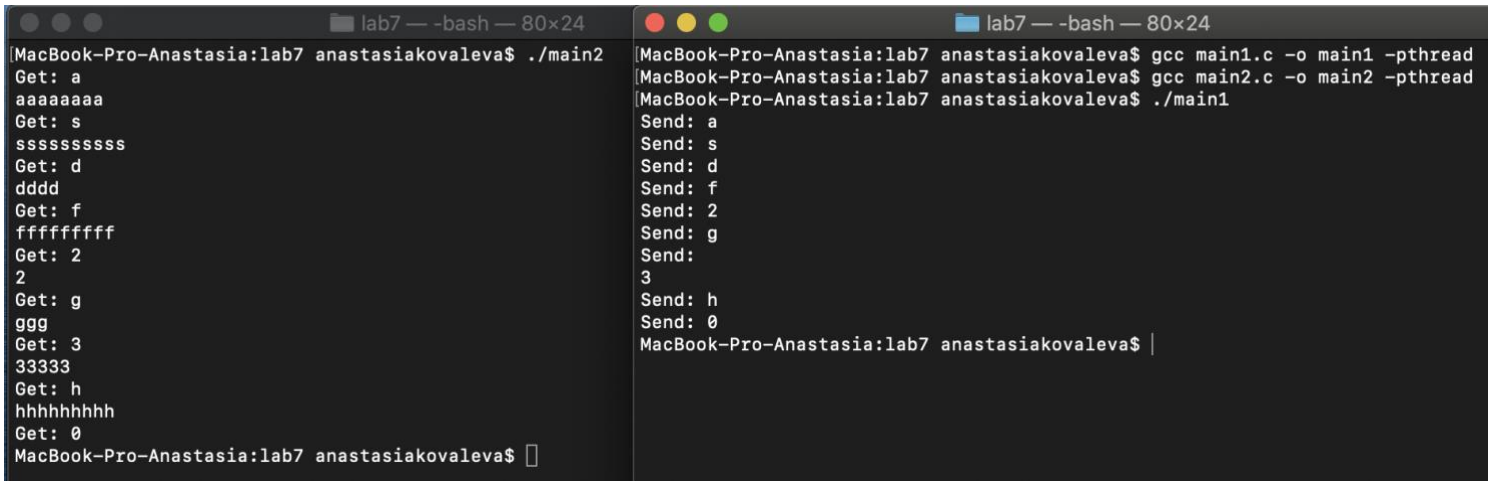
```
    char symbol;
    while(symbol != '0') {
        printf("Send: ");
        scanf("%s", &symbol);
        sem_wait(sem1);

        f = open("file", O_WRONLY | O_CREAT, 0666);

        if(f < 0) {
            printf("Can\'t open file\n");
            exit(-1);
        }
        pthread_mutex_lock(&mutex);
        write(f, &symbol, sizeof(symbol));
        close(f);
        pthread_mutex_unlock(&mutex);

        sem_post(sem1);
        sem_post(sem2);
    }

    sem_close(sem1);
    sem_close(sem2);
    return 0;
}
```

main2.c

```
#include<stdio.h>
#include<fcntl.h>
#include<sys/types.h>
#include<unistd.h>
#include<stdlib.h>
#include<semaphore.h>
#include <pthread.h>

static pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

int main() {
    sem_t *sem1,*sem2;
    int f;

    //sem_unlink("semaphore2");

    if((sem1 = sem_open("semaphore1", O_RDWR)) == SEM_FAILED) {
        printf("ERROR sem_open");
        return 1;
    }

    if((sem2=sem_open("semaphore2", O_RDWR|O_CREAT, 0666,1)) ==
SEM_FAILED) {
        printf("ERROR sem_open");
        return 1;
    }
```

```
    char symbol[2];
    while(symbol[0] != '0'){

        sem_wait(sem2);
        sem_wait(sem1);

        f = open("file",O_RDONLY);

        if(f < 0) {
            printf("Can\'t open file\n");
            exit(-1);
        }

        pthread_mutex_lock(&mutex);
        read(f, symbol, sizeof(symbol));
        printf("Get: %s \n", symbol);
        close(f);
        pthread_mutex_unlock(&mutex);

        sem_post(sem1);

        if(symbol[0] == '0') {
            break;
        }

        int times = rand()%10 + 1;
        for( int i = 0; i < times; i++) {
            printf("%s",symbol);
        }
        printf("\n");
    }

    sem_close(sem1);
    sem_close(sem2);
    return 0;
}
```

Результат выполнения:



Вывод:

Ознакомилась с реализацией семафоров в ОС Linux.