Государственное учреждение образования
“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №3
по курсу “Операционные системы”
Алгоритмы синхронизации процессов

Выполнили:
студенты группы 121702
Витковская С.И.
Шершень К.А.

Проверил:
Цирук В.А.

Минск, 2022

Цель работы: изучить алгоритмы синхронизации процессов и средства операционной системы, позволяющие осуществлять синхронизацию.

Задание: написать программу, копирующую файл. Программа должна запускать два дочерних процесса, один читает файл, другой пишет. Передача данных между процессами должна быть реализована через общую область памяти, синхронизация – с помощью объектов ядра. Материнский процесс должен обрабатывать ошибки в дочерних процессах (например, если при записи произошла ошибка – читающий процесс должен завершаться материнским).

Выполнение задания: выбранный метод синхронизации - мьютекс.

Мьютекс предназначен для организации взаимоисключающего доступа к общим данным для нескольких потоков. Поток запрашивает монопольное использование общих данных, защищаемых мьютексом, в нашем случае — участок памяти, в котором находятся данные из копируемого файла. Мы используем блокировки мьютекса для защиты области и предотвращения гонки.

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <stdlib.h>
#include <pthread.h>
#include <signal.h>
void kill_childPID(pid_t pid)
{
            kill(pid,SIGKILL);

}

int main(int argc, char *argv[])
{
    pthread_mutex_t mutex;
        pthread_mutex_init(&mutex, NULL);
        int counter;

    if(argc == 3) {

        pid_t cpid_1 , cpid_2;
        cpid_1 = fork();
        static char  *mystring;
        mystring = mmap(NULL, sizeof *mystring, PROT_READ | PROT_WRITE,
                    MAP_SHARED | MAP_ANONYMOUS, -1, 0);

        FILE *INPUT, *OUTPUT;

        INPUT = fopen(argv[1], "r");
        OUTPUT = fopen(argv[2], "w+");
        if (cpid_1 == 0){
             pthread_mutex_lock(&mutex);
            printf("child process for read\n");
            while(fgets(mystring, 10000, INPUT)) {
                puts(mystring);
                printf("read is \n");
                printf("PID is %d\n",getpid());

            };

         pthread_mutex_unlock(&mutex);


        }else{
```

```
           cpid_2 = fork();

        }
        if (cpid_2 == 0)
        {
             pthread_mutex_lock(&mutex);
            printf("child process for write \n" );
            printf("PID is %d\n",getpid());
            while(fgets(mystring, 10000, INPUT)){
                printf("write is \n");
                fputs(mystring, OUTPUT);
                 pthread_mutex_unlock(&mutex);
            }
        }else if (cpid_1 != 0){
            printf("Parent process \n");
            printf("PID is %d\n",getpid());
            if (cpid_2 < 0 )
            {
               kill_childPID(cpid_2);
            }SS
        }


    }else if(argc > 3) {
        perror("Too many arguments supplied.\n");
        exit(0);
    }
    else if(argc < 3) {
        perror("Too few arguments supplied.\n");
        exit(0);
    }

    pthread_mutex_destroy(&mutex);

    return 0;
}
```

Вывод:

```
kl1rik@kl1rik-VirtualBox:~/kl1rik$ gcc lab3mod.c -o t
kl1rik@kl1rik-VirtualBox:~/kl1rik$ ./t testlab.txt testalabcopy.txt
child process for read
defwEFKLWNFJLNWRFBWWBRWKJRBFKWRBFKHQWBFKWRBFWRKHB

read is
PID is 16377
RGWRGWRW

read is
PID is 16377
QWFWEFWEF

read is
PID is 16377
child process for write
PID is 16377
Parent process
PID is 16376
child process for write
PID is 16378
write is
write is
write is
```

Вывод: изучили алгоритмы синхронизации процессов и средства операционной системы, позволяющие осуществлять синхронизацию, подробно разобрали работу мьютекса, использовав его как основной метод синхронизации при выполнении индивидуального задания, изучили теорию и ответили на контрольные вопросы.