Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №7

за 4 семестр

По дисциплине: «ОСиСП»

Тема: «Семафоры»

Выполнила:

Студентка 2 курса

Группы ПО-3(1

Давидюк Ю. И.

Выполнила:

Студентка 2 курса

Группы ПО-3(1)

Гаврилюк Р.И.

Проверил:

Давидюк Ю. И

Выполнил:

Студент 2 курса

Группы ПО-4(2)

Кречко К.А.

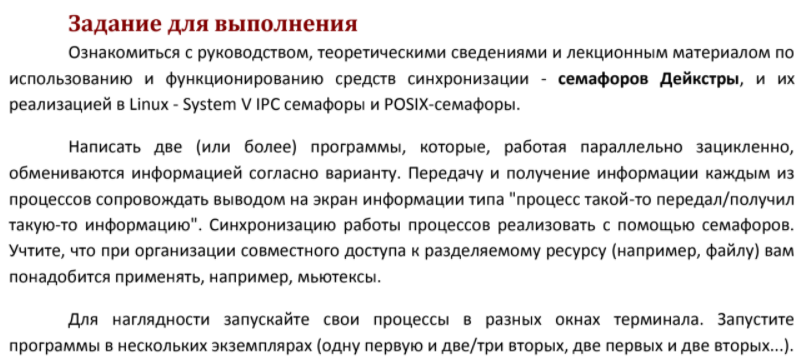
Проверила:

Давидюк Ю.И.

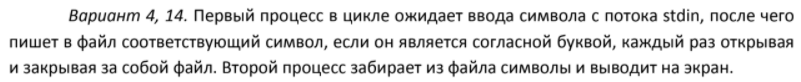
Брест 2021

Лабораторная работа №7

«Семафоры»



**Вариант – 14**

****

**p.c (первый файл – производитель)**

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/shm.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <pthread.h>

#include <string.h>

#include <semaphore.h>

#define CONSONANTS "BCDFGHJKLMNPQRSTVWXZ"

static pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

int main() {

char sym[2];

int fd;

int sizeWrite;

sem\_t \*empty = sem\_open("empty", O\_CREAT, 0777, 1);

sem\_t \*full = sem\_open("full", O\_CREAT, 0777, 0);

(void)umask(0);

while(1) {

printf("Введите символ: ");

scanf("%s", sym);

if((strchr(CONSONANTS, toupper(sym[0]))) != '\0'){

pthread\_mutex\_lock(&mutex);

if((fd = open("File", O\_RDWR | O\_CREAT, 0777)) < 0){

printf("Can\'t open file\n");

return(-1);

}

sizeWrite = write (fd, &sym[0], strlen(sym));

if(sizeWrite <= 0){

printf("Error.\n");

return(-1);

}

printf("процесс (pid=%d) отправил символ");

close(fd);

pthread\_mutex\_unlock(&mutex);

sem\_post(full);

}else{

printf("Не согласная буква. Запись не произведена\n");

continue;

}

}

pthread\_mutex\_destroy(&mutex);

sem\_close(empty);

sem\_close(full);

return 0;

}

**c.c (второй файл – потребитель )**

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/shm.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <pthread.h>

#include <semaphore.h>

static pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

int main() {

char sym;

int fd;

int sizeRead;

sem\_t \*empty = sem\_open("empty", O\_CREAT, 0777, 0);

sem\_t \*full = sem\_open("full", O\_CREAT, 0777, 0);

(void)umask(0);

while(1) {

sem\_wait(full);

pthread\_mutex\_lock(&mutex);

if((fd = open("File", O\_RDONLY | O\_CREAT, 0777)) < 0){

printf("Can\'t open file\n");

return(-1);

}

sizeRead = read(fd, &sym, sizeof(char));

if(sizeRead < 0){

printf("Can't read\n");

return(-1);

}

printf("процесс (pid=%d) принял: %s\n", getpid(), &sym);

close(fd);

pthread\_mutex\_unlock(&mutex);

sem\_post(empty);

}

pthread\_mutex\_destroy(&mutex);

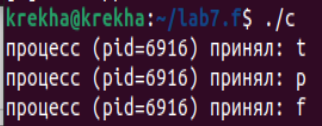
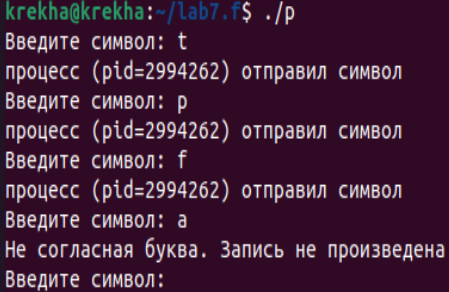
sem\_close(empty);

sem\_close(full);

return 0;

}

**Результат:**



**Вывод:** Я изучил работу с средствами межпроцессного взаимодействия в ОС Linux.