Министерство образования Республики Беларусь Учреждение образования

«Брестский государственный технический университет» Кафедра ИИТ

Лабораторная работа №6

по дисциплине «Операционные системы»

Тема«Средства межпроцессного взаимодействия»

Вариант 4

Выполнил: 220203

студент 2-го курса

группы ПО-11

Антонюк Н.А.

Проверила:

Давидюк Ю.И.

Брест 2024

**Цель работы:** Ознакомиться с руководством, теоретическими сведениями и лекционным материалом по

использованию и функционированию средств синхронизации - семафоров Дейкстры, и их

реализацией в Linux - System V IPC семафоры и POSIX-семафоры.

Написать две (или более) программы, которые, работая параллельно зацикленно,

обмениваются информацией согласно варианту. Передачу и получение информации каждым

из процессов сопровождать выводом на экран информации 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 мьютексы.

Для наглядности запускайте свои процессы в разных окнах терминала. Запустите

программы в нескольких экземплярах (одну первую и две/три вторых, две первых и две

вторых...).

**Ход работы**

***Задание для выполнения:***

Вариант 4. Первый процесс в цикле ожидает ввода символа с потока stdin, после чего

пишет в файл соответствующий символ, если он является согласной буквой, каждый раз

открывая и закрывая за собой файл. Второй процесс забирает из файла символы и выводит на

экран..

***Текст программы:***

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/types.h>

#include <fcntl.h>

pthread\_mutex\_t mutex;

FILE \*file;

void\* producer(void\* arg) {

while (1) {

char input\_char;

printf("Введите символ: ");

scanf(" %c", &input\_char);

pthread\_mutex\_lock(&mutex);

file = fopen("shared\_file.txt", "a"); // Используйте "a" для дописывания в файл

if (file != NULL) {

fprintf(file, "%c\n", input\_char);

fclose(file);

}

pthread\_mutex\_unlock(&mutex);

sleep(1); // Добавлено для удобства ввода

}

return NULL;

}

int main() {

pthread\_t producer\_thread;

pthread\_mutex\_init(&mutex, NULL);

if (pthread\_create(&producer\_thread, NULL, producer, NULL) != 0) {

perror("Error creating producer thread");

exit(EXIT\_FAILURE);

}

// Ожидаем завершения потока (это не должно произойти)

pthread\_join(producer\_thread, NULL);

pthread\_mutex\_destroy(&mutex);

return 0;

}

***Процесс 2(consumer.c):***

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/types.h>

#include <fcntl.h>

pthread\_mutex\_t mutex;

FILE \*file;

void\* consumer(void\* arg) {

while (1) {

char read\_char;

pthread\_mutex\_lock(&mutex);

file = fopen("shared\_file.txt", "r"); // Используйте "r" для чтения из файла

if (file != NULL) {

while (fscanf(file, " %c", &read\_char) == 1) {

printf("Прочитан символ: %c\n", read\_char);

}

fclose(file);

}

pthread\_mutex\_unlock(&mutex);

sleep(1); // Добавлено для удобства ввода

}

return NULL;

}

int main() {

pthread\_t consumer\_thread;

pthread\_mutex\_init(&mutex, NULL);

if (pthread\_create(&consumer\_thread, NULL, consumer, NULL) != 0) {

perror("Error creating consumer thread");

exit(EXIT\_FAILURE);

}

// Ожидаем завершения потока (это не должно произойти)

pthread\_join(consumer\_thread, NULL);

pthread\_mutex\_destroy(&mutex);

return 0;

***Вывод в консоли:***



