**Задание на работу**

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 Nкратную запись строки в файл. Читатель заканчивает работу после прочтения текущего содержимого файла. Синхронизация процессов должна выполняться с помощью семафоров.

**w.c**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <unistd.h>

#include <wait.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/sem.h>

#define N 124

int main(int argc, char \*\*argv) {

int sem\_Write; // семафор доступа к записи

int sem\_writersNow; // семафор - число готовых писателей

int sem\_readersNow; // семафор - число готовых читателей

int sem\_countProcNow; // семафор - число запущенных процессов

struct sembuf

op1 = {0, 1, 0}, // увеличение на 1

op2 = {0, -1, 0}, // уменьшение на 1

op3 = {0, 0, IPC\_NOWAIT}; // сравнение с 0

FILE \*fpOut; // файл для записи данных

// создание семафора доступа к записи

sem\_Write = semget(100 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_Write != -1){

fpOut = fopen("textOut.txt","w");

fclose(fpOut);

semop(sem\_Write, &op1, 1);

}else{

sem\_Write = semget(100 , 1, IPC\_CREAT);

if(sem\_Write == -1){

puts("Не удается подключиться к семафору доступа к записи");

return 1;

}

}

// создание семафора - число готовых писателей

sem\_writersNow = semget(200 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_writersNow == -1) {

sem\_writersNow = semget(200 , 1, IPC\_CREAT);

if(sem\_writersNow == -1){

puts("Не удается подключиться к семафору - числу готовых писателей");

return 2;

}

}

// создание семафора - число готовых читателей

sem\_readersNow = semget(300 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_readersNow == -1) {

sem\_readersNow = semget(300 , 1, IPC\_CREAT);

if(sem\_readersNow == -1){

puts("Не удается подключиться к семафору - числу готовых читателей");

return 3;

}

}

// создание семафора - число запущенных процессов

sem\_countProcNow = semget(400 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_countProcNow == -1) {

sem\_countProcNow = semget(400 , 1, IPC\_CREAT);

if(sem\_countProcNow == -1){

puts("Не удается подключиться к семафору - числу запущенных процессов");

return 4;

}

}

// увеличение числа запущенных процессов

semop(sem\_countProcNow, &op1, 1);

char str[50];

int cnt = 0;

while (cnt < \*argv[2] - '0') {

// увеличение числа готовых писателей

semop(sem\_writersNow, &op1, 1);

// проверка на наличие готовых читателей

semop(sem\_readersNow, &op3, 1);

semop(sem\_Write, &op2, 1);

// начало критической секции

fpOut = fopen("textOut.txt","a");

sprintf(str,"%s %d, %s %d" , "Писатель ", getpid(), "строка ", cnt + 1);

fprintf(fpOut, "%s\n", str);

puts(str);

fclose(fpOut);

// конец критической секции

semop(sem\_Write, &op1, 1);

// уменьшение числа готовых писателей

semop(sem\_writersNow, &op2, 1);

sleep(\*argv[1] - '0');

cnt++;

}

// уменьшение числа запущенных процессов

semop(sem\_countProcNow, &op2, 1);

if (semop(sem\_countProcNow, &op3, 1) != -1) {

semctl(sem\_Write, 0 , IPC\_RMID, 0);

semctl(sem\_writersNow, 0 , IPC\_RMID, 0);

semctl(sem\_readersNow, 0 , IPC\_RMID, 0);

semctl(sem\_countProcNow, 0 , IPC\_RMID, 0);

}

return 0;

}

**r.c**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <unistd.h>

#include <wait.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/sem.h>

#define N 124

int main(int argc, char \*\*argv) {

int sem\_Write; // семафор доступа к записи

int sem\_writersNow; // семафор - число готовых писателей

int sem\_readersNow; // семафор - число готовых читателей

int sem\_countProcNow; // семафор - число запущенных процессов

struct sembuf

op1 = {0, 1, 0}, // увеличение на 1

op2 = {0, -1, 0}, // уменьшение на 1

op3 = {0, 0, IPC\_NOWAIT}; // сравнение с 0

FILE \*fpIn; // файл для записи данных

// создание семафора доступа к записи

sem\_Write = semget(100 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_Write != -1){

semop(sem\_Write, &op1, 1);

}else{

sem\_Write = semget(100 , 1, IPC\_CREAT);

if(sem\_Write == -1){

puts("Не удается подключиться к семафору доступа к записи");

return 1;

}

}

// создание семафора - число готовых писателей

sem\_writersNow = semget(200 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_writersNow == -1) {

sem\_writersNow = semget(200 , 1, IPC\_CREAT);

if(sem\_writersNow == -1){

puts("Не удается подключиться к семафору - числу готовых писателей");

return 2;

}

}

// создание семафора - число готовых читателей

sem\_readersNow = semget(300 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_readersNow == -1) {

sem\_readersNow = semget(300 , 1, IPC\_CREAT);

if(sem\_readersNow == -1){

puts("Не удается подключиться к семафору - числу готовых читателей");

return 3;

}

}

// создание семафора - число запущенных процессов

sem\_countProcNow = semget(400 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_countProcNow == -1) {

sem\_countProcNow = semget(400 , 1, IPC\_CREAT);

if(sem\_countProcNow == -1){

puts("Не удается подключиться к семафору - числу запущенных процессов");

return 4;

}

}

// увеличение числа запущенных процессов

semop(sem\_countProcNow, &op1, 1);

// проверка на наличие готовых писателей

semop(sem\_writersNow, &op3, 1);

// увеличение числа готовых читателей

semop(sem\_readersNow, &op1, 1);

// начало критической секции

char str[50];

fpIn = fopen("textOut.txt","r");

while (fgets(str, 50, fpIn) != NULL) {

str[strlen(str) - 1] = '\0';

puts(str);

}

fclose(fpIn);

// конец критической секции

// уменьшение числа готовых читателей

semop(sem\_readersNow, &op2, 1);

// уменьшение числа запущенных процессов

semop(sem\_countProcNow, &op2, 1);

if (semop(sem\_countProcNow, &op3, 1) != -1) {

semctl(sem\_Write, 0 , IPC\_RMID, 0);

semctl(sem\_writersNow, 0 , IPC\_RMID, 0);

semctl(sem\_readersNow, 0 , IPC\_RMID, 0);

semctl(sem\_countProcNow, 0 , IPC\_RMID, 0);

}

return 0;

}