|  |
| --- |
| «Санкт-Петербургский государственный электротехнический университет |
| «ЛЭТИ» им. В.И.Ульянова(Ленина)» |
| (СПбГЭТУ «ЛЭТИ») Кафедра вычислительной техники |
|  |
| **Отчет**  **по лабораторным работам № 10**  **по дисциплине** **«Организация процессов и программирование в среде Linux»** |
| Тема: «Cинхронизация процессов с помощью семафоров» |
|  |

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Студент гр. 9307 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_Брызгалова Е. А. |
|  |  |
|  |  |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Разумовский Г. В. |

Оглавление

[Цель работы: 2](#_Toc118034666)

[Задание 2](#_Toc118034667)

[Распечатки входных и выходного файлов. 2](#_Toc118034668)

[Текст программы main.cpp 3](#_Toc118034669)

[Вывод 6](#_Toc118034670)

Цель работы:

Знакомство с организацией семафоров, системными функциями, обеспечивающими управление семафорами, и их использованием для решения задач взаимоисключения и синхронизации.

Задание

1) 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 N-кратную запись строки в файл. Читатель заканчивает работу после прочтения текущего содержимого файла. Синхронизация процессов должна выполняться с помощью семафоров

2) Откомпилировать все программы и запустить их

Скриншоты работы каждой программы.

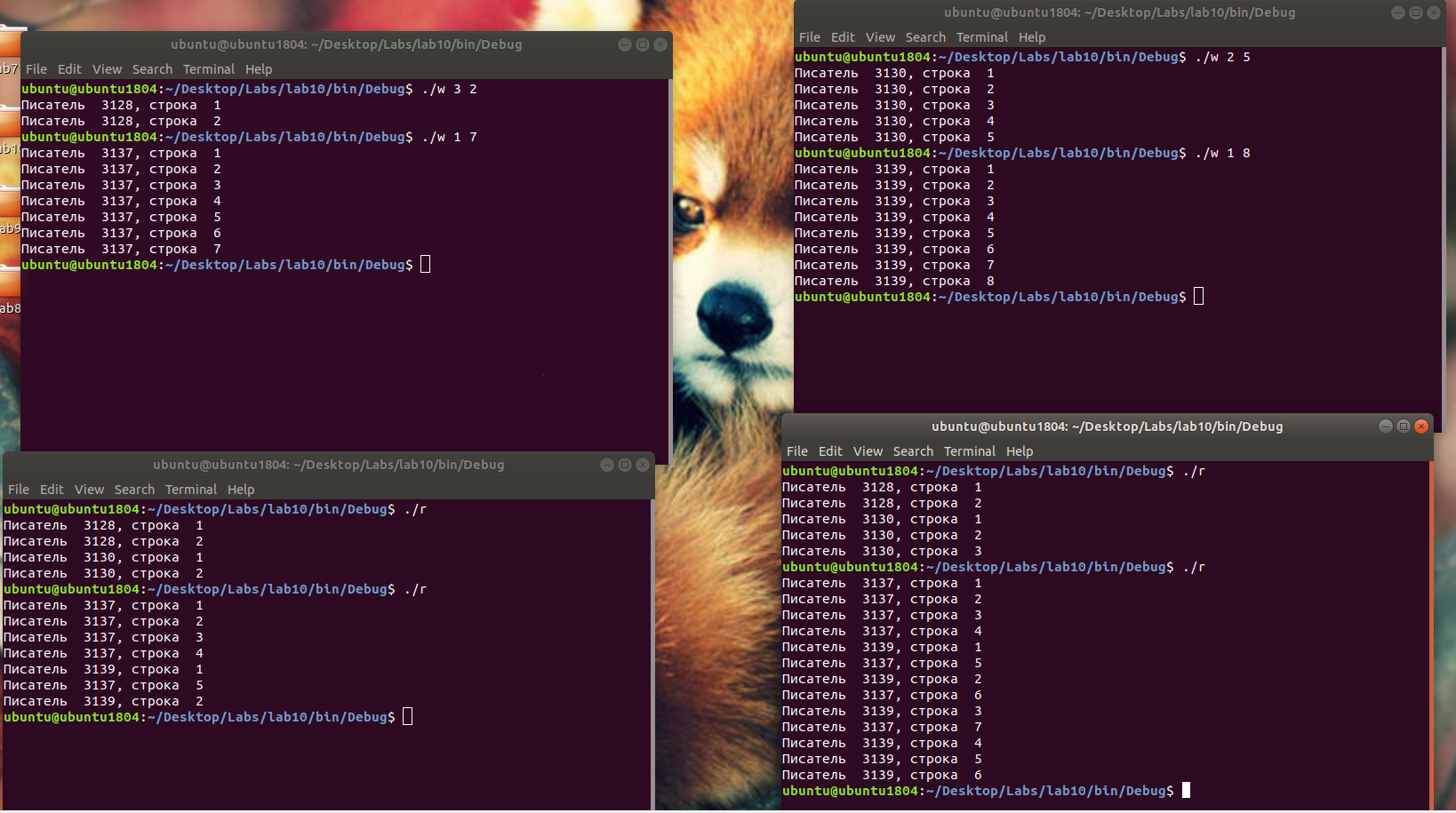


Рисунок 1. Работа программ

# **Текст программы writer**

#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>

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

int sem\_Write; // семафор доступа к записи

int sem\_writersNow; // семафор - число писателей

int sem\_readersNow; // семафор - число читателей

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

struct sembuf//Список команд {индекс простого семафора,тип команды,}

inc = {0, 1, 0},

dec = {0, -1, 0},

zero = {0, 0, 0},

fin = {0, 0, IPC\_NOWAIT};

FILE \*fpOut; // файл для записи данных

// создание семафора доступа к записи

{

sem\_Write = semget(111 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_Write != -1){

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

fclose(fpOut);

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

}else{

sem\_Write = semget(111, 1, IPC\_CREAT);

if(sem\_Write == -1){

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

return 1;

}

}

}

// создание семафора - число писателей

{

sem\_writersNow = semget(222 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_writersNow == -1) {

sem\_writersNow = semget(222 , 1, IPC\_CREAT);

if(sem\_writersNow == -1){

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

return 2;

}

}

}

// создание семафора - число читателей

{

sem\_readersNow = semget(333 , 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 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, &inc, 1);

char str[50];

int cnt = 0;

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

// увеличение числа писателей

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

// проверка на наличие читателей

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

//ожидание, захват семофора

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

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

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

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

puts(str);

fclose(fpOut);

//освобождение семофора

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

// уменьшение числа готовых писателей

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

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

cnt++;

}

// уменьшение числа запущенных процессов

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

// проверка на наличие процессов, уничтожение семафоров

if (semop(sem\_countProcNow, &fin, 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;

}

# **Текст программы reader**

#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>

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

int sem\_Write; // семафор доступа к записи

int sem\_writersNow; // семафор - число писателей

int sem\_readersNow; // семафор - число читателей

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

struct sembuf

inc = {0, 1, 0},

dec = {0, -1, 0},

zero = {0, 0, 0},

fin = {0, 0, IPC\_NOWAIT};

FILE \*fpIn; // файл для записи данных

// создание семафора доступа к записи

{

sem\_Write = semget(111 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_Write != -1){

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

}else{

sem\_Write = semget(111 , 1, IPC\_CREAT);

if(sem\_Write == -1){

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

return 1;

}

}

}

// создание семафора - число писателей

{

sem\_writersNow = semget(222 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_writersNow == -1) {

sem\_writersNow = semget(222 , 1, IPC\_CREAT);

if(sem\_writersNow == -1){

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

return 2;

}

}

}

// создание семафора - число читателей

{

sem\_readersNow = semget(333 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_readersNow == -1) {

sem\_readersNow = semget(333 , 1, IPC\_CREAT);

if(sem\_readersNow == -1){

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

return 3;

}

}

}

// создание семафора - число запущенных процессов

{

sem\_countProcNow = semget(444 , 1, 0606 | IPC\_CREAT | IPC\_EXCL);

if(sem\_countProcNow == -1) {

sem\_countProcNow = semget(444 , 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, &inc, 1);

// проверка на наличие писателей

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

// увеличение числа читателей

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

char str[50];

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

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

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

puts(str);

}

fclose(fpIn);

// уменьшение числа читателей

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

// уменьшение числа запущенных процессов

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

// проверка на наличие процессов, уничтожение семафоров

if (semop(sem\_countProcNow, &fin, 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;

}

Вывод

В процессе выполнения лабораторной работы познакомились с сорганизацией семафоров, системными функциями, обеспечивающими управление семафорами, и их использованием для решения задач взаимоисключения и синхронизации.