Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

Институт электронной техники и приборостроения

Кафедра Информационная безопасность автоматизированных систем

Специальность 10.05.03 Информационная безопасность автоматизированных систем

Практическая работа № 4 по дисциплине «Операционная система UNIX»

Тема «Семафоры в OC UNIX»

|  |  |
| --- | --- |
|  | Выполнил: студент 3 курса  учебной группы с1-ИБС 32  очной формы обучения  Солодилов В.В.  Проверила: профессор  кафедры ИБС  Пластун И. Л. |

Саратов 2022

**Задание**

Процесс 1 порождает потомков 2 и 3, все они присоединяют к себе разделяемую память объёмом (2 \* sizeof(int)). Процессы 1 и 2 по очереди пишут в эту память число, равное своему номеру (1 или 2). После этого один из процессов удаляет разделяемую память, затем процесс 3 считывает содержимое области разделяемой памяти и записывает в файл. Используя семафоры, обеспечить следующее содержимое файла:

б) 1 1 2 2 1 1 2 2

**Листинг программы**

#include <sys/ipc.h>

#include <sys/sem.h>

#include <sys/shm.h>

#include <stdio.h>

#include <sys/stat.h>

#include <unistd.h>

#include <stdlib.h>

#include <stdbool.h>

#define SEM\_ID 2001 //ключ массива семафоров

#define SHM\_ID\_ARR 2002 //ключ разделяемой памяти

int main() {

FILE \*file = fopen("text.txt", "w+"); //открываем файл для записи

int \*array; //массив 2 \* int

int pid = getpid();//инициализация 1-го потока

int pid2, pid3; //создание 2-го и 3-го потока

int number = 1; //записываемое число

int writeRegister = 0; //число записей в текстовый документ

//IPC\_CREAT - создание новой очередей сообщений с ключом, 0666 - чтение и запись для всех пользователей и групп

//int semget(key\_t key, int nsems, int semflg) - считывает идентификатор набора семафоров и получает доступ к ним

int semid = semget(SEM\_ID, 1, IPC\_CREAT | 0666);

//int shmget (IPC\_PRIVATE, SEGMENT\_SIZE, IPC\_CREAT | IPC\_EXCL | S\_IRUSR | S\_IWUSR) - присваивает идентификатор разделяемому сегменту памяти

int shmid = shmget(SHM\_ID\_ARR, 2 \* sizeof(int), IPC\_CREAT | 0666); //выделение 2 int общей памяти

array = (int \*) shmat(shmid, 0, 0); //присоединение области разделяемой памяти к виртуальному адресному пространству

semctl(semid, 0, SETVAL, 0);//инициализация семафоровов

//инициализация 2-го и 3-го дочерних потоков

pid2 = fork();

pid3 = fork();

while (true) {

if (semctl(semid, 0, GETVAL, 0)) continue; // ожидание процессов

semctl(semid, 0, SETVAL, 1); // установка значения на выполнение процессов

//switch для записи данных в файл

switch (number) {

case 1: {

printf("Process %d (%d) write %d \n", number, pid, number);

array[0] = number;

number++;

}

case 2: {

printf("Process %d (%d) write %d \n", number, pid2, number);

array[1] = number;

}

case 3: {

//Определение записываемых чисел

switch (writeRegister) {

case 0: case 2: {

printf("Process 3 (%d) write in the file: %d и %d\n\n", pid3, array[0], array[0]);

fprintf(file, "%d%d", array[0], array[0]); //запись

break;

}

case 1: case 3: {

printf("Process 3 (%d) write in the file: %d и %d\n\n", pid3, array[1], array[1]);

fprintf(file, "%d%d", array[1], array[1]); //запись

break;

}

}

//обнуление разделяемой памяти

array[0] = 0;

array[1] = 0;

writeRegister++; //увеличение счётчика записанных данных

switch(writeRegister) {

case 4: {

shmdt(array); //очистка массива int

exit(0); //завершение процессов

} default: {

number = 1; //возврат числа к исходному значению

semctl(semid, 0, SETVAL, 0); //установка на блокировку

}

}

}

}

}

}

**Результаты выполнения программы**

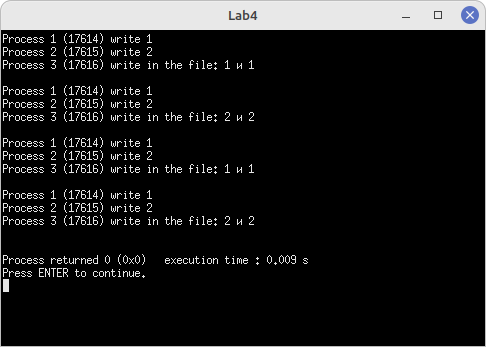


Рисунок 1 — Результат выполнения программы

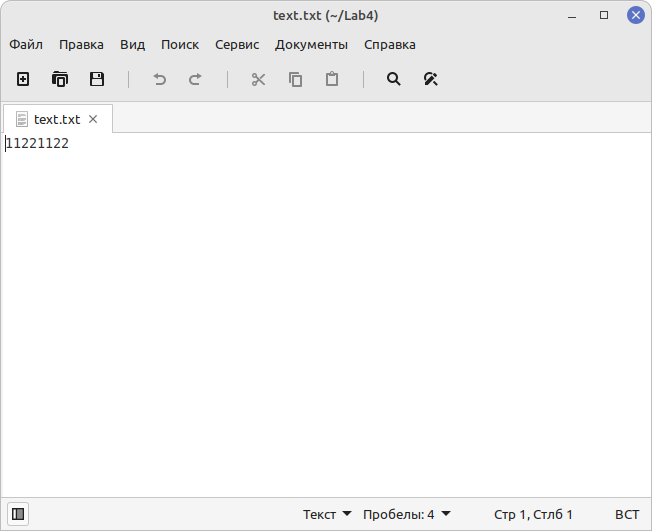


Рисунок 2 — Конечный файл