Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ ПО ДИСЦИПЛИНЕ

«Операционная система Linux»

ЛАБОРАТОРНАЯ РАБОТА №10

«Семафоры и разделяемая память»

Выполнили студенты группы 19ВВ2:

Гусев Д.О.

Кубасов И.М.

Приняли:

Федюнин Р.Н.

Семенов А.О.

Пенза 2022

**Цель работы :** Изучение механизмов взаимодействия процессов посредством разделяемой памяти и синхронизации с использованием семафоров.

Вариант 2.

**Задания:** Запустить 5 параллельных процессов (использовать fork), и одновременно во всех запустить цикл от 1 до 30 со случайной задержкой внутри цикла. Синхронизировать итерации циклов (1я итерация процессов, затем 2я итерация процессов и т.д.). Все процессы должны завершиться одновременно.

**Листинг программы:**

#include <iostream>

#include <cstdlib>

#include <sys/stat.h>

#include <fcntl.h>

#include <signal.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <unistd.h>

#include <string.h>

#include <sys/shm.h>

#include <iostream>

#include <sys/sem.h>

#include <ctime>

using namespace std;

#define Proj\_Id 13

int IDShm=0;

int IDSemaphore=0;

const char\* MSG\_Memory = "/tmp/LabMemory30";

union semun

{

int val;

struct semid\_ds \*buf;

ushort array [25];

} arg;

int main(int argc,char \*argv[])

{

unsigned nsops=1; //аргумент необходим для semop (указывате кол-во операций), без него не работает

srand( time(0) );

sembuf \*SemophoreBuf=new sembuf;

int IDProc=1;

unsigned long MircoSeconds=0;

int \*SHMemory=0;

key\_t key;

pid\_t pid;

int file = open(MSG\_Memory, O\_CREAT | O\_RDWR | O\_APPEND, 0755);

close(file);

key=ftok(MSG\_Memory, Proj\_Id);

if(key==-1)

cout<<"Error Ftok"<<endl;

IDShm=shmget(key,1024,IPC\_CREAT|0660); //0660 необходим для разрешения записи и чтения права доступа

if(IDShm==-1)

cout<<"Error shmget"<<endl;

SHMemory=(int\*)shmat(IDShm,NULL,0);

IDSemaphore=semget(key,6,IPC\_CREAT|0660);

if (SHMemory[0]==0)

{

semctl (IDSemaphore, 1, SETVAL,5);

SHMemory[0]=rand() % 30 - 1;

SHMemory[1]=1;

}

for (int i = 0; i < 4; ++i)

{

pid = fork();

if (pid > 0)

{

continue;

}

else if (pid == 0) /\* I am a child, get to work \*/

{

break;

}

else

{

printf("fork error\n");

exit(1);

}

}

IDProc=SHMemory[1];

SHMemory[1]=SHMemory[1]+1;

srand( IDProc );

MircoSeconds=rand() % 1000- 50;

SemophoreBuf->sem\_num=1;

SemophoreBuf->sem\_flg=0660;

for (int i=0; i<SHMemory[0]; i++)

{

usleep(MircoSeconds);

cout<<"Iteration:"<<i<<" Proc ID:"<<IDProc<<endl;

SemophoreBuf->sem\_op=-1;

semop(IDSemaphore,SemophoreBuf, nsops);

SemophoreBuf->sem\_op=0;

semop(IDSemaphore,SemophoreBuf, nsops);

SemophoreBuf->sem\_op=1;

semop(IDSemaphore,SemophoreBuf, nsops);

}

cout<<"Exit Proc ID:"<<IDProc<<endl;

}**Результат работы программы:**

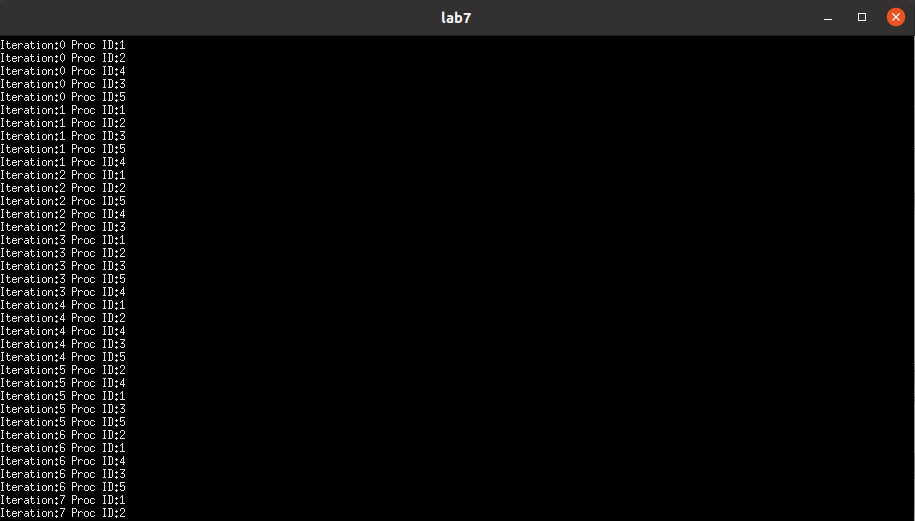


Рисунок 1— Итерации процессов (часть 1)

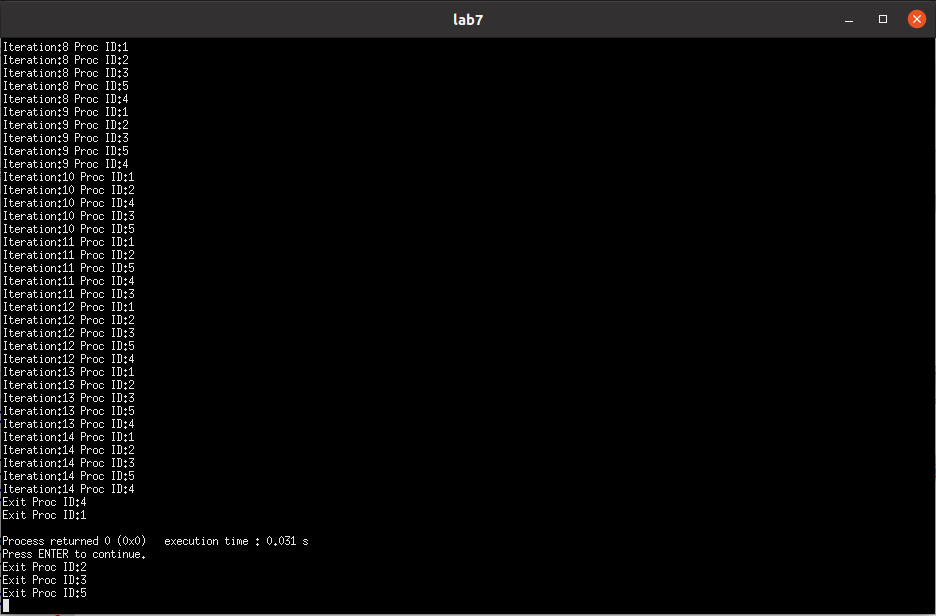


Рисунок 2— Итерации процессов (часть 2)

**Вывод:** В ходе выполнения лабораторной работы были изучены механизмы взаимодействия процессов посредством разделяемой памяти и синхронизации с использованием семафоров.