## МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

## «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**Кафедра Вычислительной техники**

## ОТЧЕТ

**по лабораторной работе №9**

## по дисциплине «Организация процессов и программирование в среде Linux»

**Тема: Обмен данными через разделяемую память**

|  |  |
| --- | --- |
| Студент гр. 8308 | Петров Г.А. |
| Преподаватель | Разумовский Г.В. |

Санкт-Петербург 2021

# Цель работы

Целью лабораторной работы является знакомство с организацией разделяемой памяти и системными функциями, обеспечивающими обмен данными между процессами.

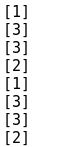
# Задание

Написать 3 программы, которые запускаются в произвольном порядке и построчно записывают свои индивидуальные данные в один файл через определенный промежуток времени. Пока не закончит писать строку одна программа, другие две не должны обращаться к файлу. Частота записи данных в файл и количество записываемых строк определяются входными параметрами, задаваемыми при запуске каждой программы. При завершении работы одной из программ другие должны продолжить свою работу. Синхронизация работы программ должна осуществляться с помощью общих переменных, размещенных в разделяемой памяти.

# Примеры выполнения программы

Программы были разработаны и откомпилированы. После чего программы были запушена через три терминала. Результаты работы программ приведены на рисунках 1-4. Так же представлена распечатка файла TEXT.txt.

TEXT.txt



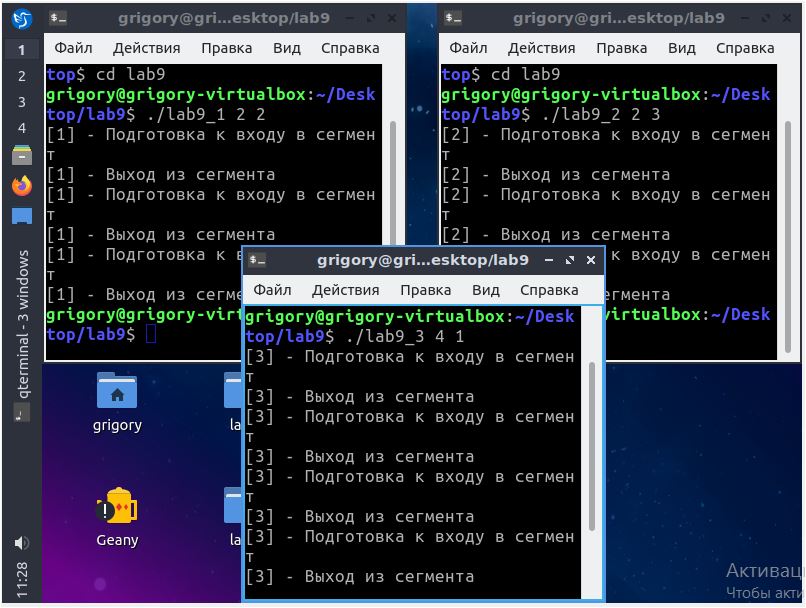


Рисунок 1

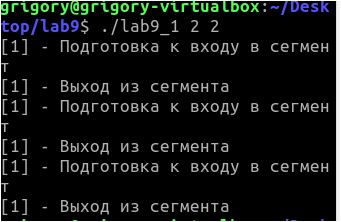


Рисунок 2

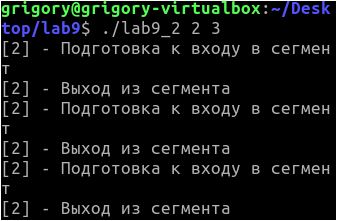


Рисунок 3

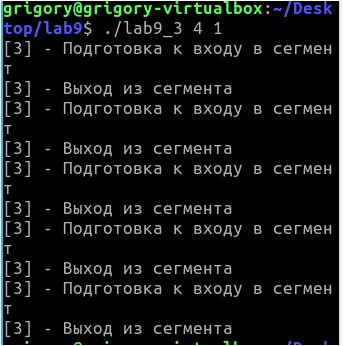


Рисунок 4

# Исходный код программ

## Lab9\_1.cpp

#include "functions.h"

#include <unistd.h>

#include <fstream>

//1 - количество, 2 - период

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

{

void\* segment\_addr=NULL;

int count\_str,period;

count\_str=atoi(argv[1]);

period=atoi(argv[2]);

//создание сигмента если не создан

if(SID==-1)

getSegment();

//добавление в адрессное пространство процесса

segment\_addr=includeSegment();

//создание указателя на общую переменную

shared\* share=(shared\*)segment\_addr;

//цикл записей

while(true)

{

sleep(period);

std::cout<<"[1] - Подготовка к входу в сегмент"<<std::endl;

lock(share,FIRST);//получение доступа

if(count\_str>0)

{//вывод данных в разделяемый файл

std::ofstream fout(FNAME,std::ios\_base::app);

fout<<"[1]"<<std::endl;

fout.close();

--count\_str;

}

else

{//сигнал окончания работы

++(share->stop);

unlock(share,FIRST);//выкл заинтересованность в ресурсе

std::cout<<"[1] - Выход из сегмента"<<std::endl;

break;//выход из цикла

}

unlock(share,FIRST);//выкл заинтересованность в ресурсе

std::cout<<"[1] - Выход из сегмента"<<std::endl;

}

if(share->stop==3)

{//если все процессы закончили работу и этот последний

excludeSegment(segment\_addr);

destructSegment();

}

return 0;

}

## Lab9\_2.cpp

#include "functions.h"

#include <unistd.h>

#include <fstream>

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

{

void\* segment\_addr=NULL;

int count\_str,period;

count\_str=atoi(argv[1]);

period=atoi(argv[2]);

//создание сигмента если не создан

if(SID==-1)

getSegment();

//добавление в адрессное пространство процесса

segment\_addr=includeSegment();

//создание указателя на общую переменную

shared\* share=(shared\*)segment\_addr;

//цикл записей

while(true)

{

sleep(period);

std::cout<<"[2] - Подготовка к входу в сегмент"<<std::endl;

lock(share,SECOND);//получение доступа

if(count\_str>0)

{//вывод данных в разделяемый файл

std::ofstream fout(FNAME,std::ios\_base::app);

fout<<"[2]"<<std::endl;

fout.close();

--count\_str;

}

else

{//сигнал окончания работы

++(share->stop);

unlock(share,SECOND);//выкл заинтересованность в ресурсе

std::cout<<"[2] - Выход из сегмента"<<std::endl;

break;//выход из цикла

}

unlock(share,SECOND);//выкл заинтересованность в ресурсе

std::cout<<"[2] - Выход из сегмента"<<std::endl;

}

if(share->stop==3)

{//если все процессы закончили работу и этот последний

excludeSegment(segment\_addr);

destructSegment();

}

return 0;

}

## Lab9\_3.cpp

#include "functions.h"

#include <unistd.h>

#include <fstream>

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

{

void\* segment\_addr=NULL;

int count\_str,period;

count\_str=atoi(argv[1]);

period=atoi(argv[2]);

//создание сигмента если не создан

if(SID==-1)

getSegment();

//добавление в адрессное пространство процесса

segment\_addr=includeSegment();

//создание указателя на общую переменную

shared\* share=(shared\*)segment\_addr;

//цикл записей

while(true)

{

sleep(period);

std::cout<<"[3] - Подготовка к входу в сегмент"<<std::endl;

lock(share,THIRD);//получение доступа

if(count\_str>0)

{//вывод данных в разделяемый файл

std::ofstream fout(FNAME,std::ios\_base::app);

fout<<"[3]"<<std::endl;

fout.close();

--count\_str;

}

else

{//сигнал окончания работы

++(share->stop);

unlock(share,THIRD);//выкл заинтересованность в ресурсе

std::cout<<"[3] - Выход из сегмента"<<std::endl;

break;//выход из цикла

}

unlock(share,THIRD);//выкл заинтересованность в ресурсе

std::cout<<"[3] - Выход из сегмента"<<std::endl;

}

if(share->stop==3)

{//если все процессы закончили работу и этот последний

excludeSegment(segment\_addr);

destructSegment();

}

return 0;

}

## Functions.h

#include <iostream>

#include <sys/shm.h>

#include <algorithm>

#define FNAME "TEXT.txt"

#define KEY 777

#define NUM\_PROCESS 3

#define FIRST 0

#define SECOND 1

#define THIRD 2

struct shared

{

int choosing[NUM\_PROCESS];//флаг ожидания выдачи номерка

int number[NUM\_PROCESS];//номерок в очереди

int stop;

};

int SID=-1;

void getSegment();//запрос на сегмент

void\* includeSegment();//включение сегмента в пространство процесса

void excludeSegment(void\*);//исключение

void destructSegment(int);//уничтожение сегмента

void lock(shared\*,int);//алгоритм пекарни доступа к сегементу

void unlock(shared\*,int);//отклчение инетерса к сегемнту

void lock(shared\* share,int proc\_i)

{

//выдача и расчет номерка

share->choosing[proc\_i]=1;

share->number[proc\_i]=1+std::max({share->number[FIRST],share->number[SECOND],share->number[THIRD]});

share->choosing[proc\_i]=0;

for(int i=0;i<NUM\_PROCESS;++i)

{

if(i!=proc\_i)

{

//ждем пока процесс i получит номерок

while(share->choosing[i]);

//ждем пока все процессы с меньшим номером закончат работу

while(share->number[i]!=0 &&

(share->number[proc\_i]>share->number[i] ||

(share->number[proc\_i]==share->number[i] &&

proc\_i>i)));

}

}

}

void unlock(shared\* share,int proc\_i)

{

share->number[proc\_i]=0;

}

void getSegment()

{

SID=shmget(KEY,sizeof(shared),0666|IPC\_CREAT);

if(SID==-1)

{

perror("error in function [shmget()]");

}

}

void\* includeSegment()

{

void\* segment\_addr=shmat(SID,NULL,0);

if(\*(int\*)segment\_addr==-1)

{

perror("error in function [shmat()]");

}

return segment\_addr;

}

void excludeSegment(void\* segment\_addr)

{

if(shmdt(segment\_addr) == -1)

{

perror("error in function [shmdt()]");

}

}

void destructSegment()

{

if(shmctl(SID,IPC\_RMID,NULL)==-1)

{

perror("error in function [shmctl()]");

}

}

# Вывод

При выполнении лабораторной работы изучены и использованы организация разделяемой памяти и системные функции, обеспечивающие обмен данными между процессами. Программа разработанная в соответствии с заданием, работает корректно.