|  |
| --- |
| «Санкт-Петербургский государственный электротехнический университет |
| «ЛЭТИ» им. В.И.Ульянова(Ленина)» |
| (СПбГЭТУ «ЛЭТИ») Кафедра вычислительной техники |
|  |
| **Отчет**  **по лабораторным работам № 9**  **по дисциплине** **«Организация процессов и программирование в среде Linux»** |
| Тема: «Обмен данными через разделяемую память» |
|  |

|  |
| --- |
|  |

|  |  |
| --- | --- |
| 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 разделяемой памяти и системными функциями, обеспечивающими обмен данными между процессами.

Задание

1) 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.

2) Откомпилировать все программы и запустить их

Распечатки входных и выходного файлов.

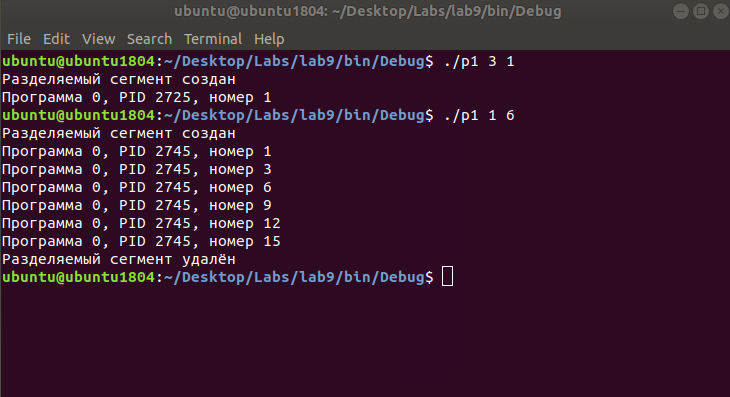


Рисунок 1. Работа программы 1

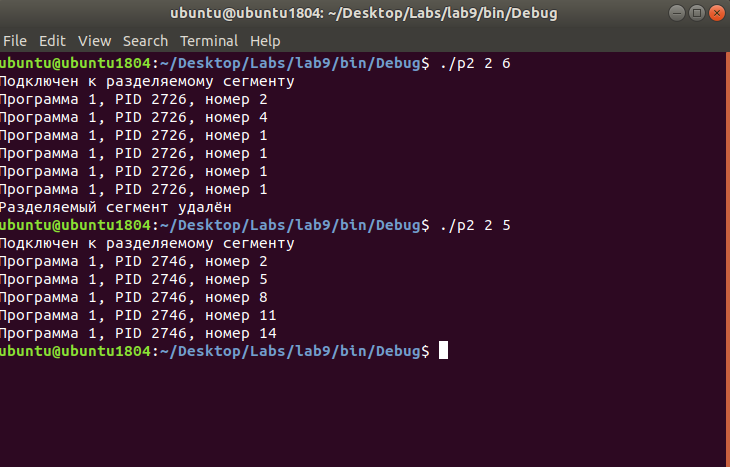


Рисунок 2. Работа программы 2

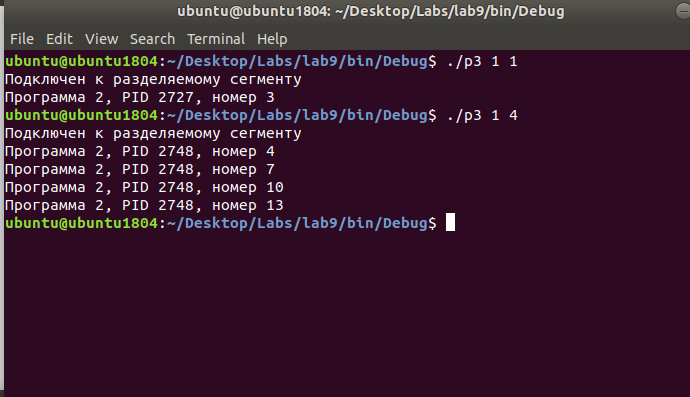


Рисунок 3. Работа программы 3

# **Текст программы 1**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <unistd.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <time.h>

#define n 3

int number[n] = {0};

bool wait\_other(int e int a, int b, int c, int d) {

return (e!=0||a < c || (a == c && b < d));

}

int max\_num(int \*a) {

int k = 0;

for (int i = 0; i < n; i++) {

if (a[i] >= k) k = a[i];

}

return k;

}

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

bool is\_creator; // метка создателя разделяемого сегмента

int prog\_id = 0; // метка программы

int shm\_id; // идентификатор разделяемого сегмента

void \*shared\_mem; // адрес привязки к процессу

FILE \*fpOut; // файл для записи данных

int pid =getpid();

//создание разделяемого сегмента

shm\_id = shmget(128, 4096, 0606 | IPC\_CREAT | IPC\_EXCL);

if(shm\_id != -1){

is\_creator = true;

puts("Разделяемый сегмент создан");

}else{

shm\_id = shmget(128, 4096, IPC\_CREAT);

if(shm\_id == -1){

puts("Не удается подключиться к разделяемому сегменту");

return 1;

}else{

puts("Подключен к разделяемому сегменту");

}

}

// присоединение сегмента к процессу

shared\_mem = shmat(shm\_id, 0, 0);

if (is\_creator) {

for (int i = 0; i < n; i++) \*(bool\*)(shared\_mem + i) = false;

for (int i = 0; i < n; i++) \*(int\*)(shared\_mem + n + 4\*i) = 0;

}

// алгоритм Лампорта

for(int i=0;i < \*argv[2] - '0';i++) {

\*(bool\*)(shared\_mem + prog\_id) = true;

memcpy(number,shared\_mem + n, 4\*n);

\*(int\*)(shared\_mem + n + 4\*prog\_id) = max\_num(number) + 1;

\*(bool\*)(shared\_mem + prog\_id) = false;

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

for(int j = 0; j < n; j++) {

while(\*(bool\*)(shared\_mem + j) ==1);

while(wait\_other(\*(int\*)(shared\_mem + n + 4\*j),\*(int\*)(shared\_mem + n + 4\*j),j,\*(int\*)(shared\_mem + n + 4\*prog\_id),prog\_id));

}

// начало критической секции

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

printf("Программа %d, PID %d, номер %d\n", prog\_id,pid, \*(int\*)(shared\_mem + n + 4\*prog\_id));

fprintf(fpOut, "Программа %d, PID %d, номер %d\n", prog\_id, pid, \*(int\*)(shared\_mem + n + 4\*prog\_id));

fclose(fpOut);

// конец критической секции

\*(int\*)(shared\_mem + n + 4\*prog\_id) = 0;

}

// отсоединение сегмента

memcpy(number,shared\_mem + n, 4\*n);

shmdt(shared\_mem);

bool f = true;

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

if(number[i] != 0)

f=false;

// удаление разделяемого сегмента

if (f) {

if (shmctl(shm\_id, IPC\_RMID, 0) == 0) puts("Разделяемый сегмент удалён");

}

return 0;

}

# **Текст программы 2**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <unistd.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <time.h>

#define n 3

int number[n] = {0};

bool wait\_other(int e int a, int b, int c, int d) {

return (e!=0||a < c || (a == c && b < d));

}

int max\_num(int \*a) {

int k = 0;

for (int i = 0; i < n; i++) {

if (a[i] >= k) k = a[i];

}

return k;

}

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

bool is\_creator; // метка создателя разделяемого сегмента

int prog\_id = 1; // метка программы

int shm\_id; // идентификатор разделяемого сегмента

void \*shared\_mem; // адрес привязки к процессу

FILE \*fpOut; // файл для записи данных

int pid =getpid();

//создание разделяемого сегмента

shm\_id = shmget(128, 4096, 0606 | IPC\_CREAT | IPC\_EXCL);

if(shm\_id != -1){

is\_creator = true;

puts("Разделяемый сегмент создан");

}else{

shm\_id = shmget(128, 4096, IPC\_CREAT);

if(shm\_id == -1){

puts("Не удается подключиться к разделяемому сегменту");

return 1;

}else{

puts("Подключен к разделяемому сегменту");

}

}

// присоединение сегмента к процессу

shared\_mem = shmat(shm\_id, 0, 0);

if (is\_creator) {

for (int i = 0; i < n; i++) \*(bool\*)(shared\_mem + i) = false;

for (int i = 0; i < n; i++) \*(int\*)(shared\_mem + n + 4\*i) = 0;

}

// алгоритм Лампорта

for(int i=0;i < \*argv[2] - '0';i++) {

\*(bool\*)(shared\_mem + prog\_id) = true;

memcpy(number,shared\_mem + n, 4\*n);

\*(int\*)(shared\_mem + n + 4\*prog\_id) = max\_num(number) + 1;

\*(bool\*)(shared\_mem + prog\_id) = false;

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

for(int j = 0; j < n; j++) {

while(\*(bool\*)(shared\_mem + j) ==1);

while(wait\_other(\*(int\*)(shared\_mem + n + 4\*j),\*(int\*)(shared\_mem + n + 4\*j),j,\*(int\*)(shared\_mem + n + 4\*prog\_id),prog\_id));

}

// начало критической секции

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

printf("Программа %d, PID %d, номер %d\n", prog\_id,pid, \*(int\*)(shared\_mem + n + 4\*prog\_id));

fprintf(fpOut, "Программа %d, PID %d, номер %d\n", prog\_id, pid, \*(int\*)(shared\_mem + n + 4\*prog\_id));

fclose(fpOut);

// конец критической секции

\*(int\*)(shared\_mem + n + 4\*prog\_id) = 0;

}

// отсоединение сегмента

memcpy(number,shared\_mem + n, 4\*n);

shmdt(shared\_mem);

bool f = true;

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

if(number[i] != 0)

f=false;

// удаление разделяемого сегмента

if (f) {

if (shmctl(shm\_id, IPC\_RMID, 0) == 0) puts("Разделяемый сегмент удалён");

}

return 0;

}

# **Текст программы 3**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <unistd.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <time.h>

#define n 3

int number[n] = {0};

bool wait\_other(int e int a, int b, int c, int d) {

return (e!=0||a < c || (a == c && b < d));

}

int max\_num(int \*a) {

int k = 0;

for (int i = 0; i < n; i++) {

if (a[i] >= k) k = a[i];

}

return k;

}

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

bool is\_creator; // метка создателя разделяемого сегмента

int prog\_id = 2; // метка программы

int shm\_id; // идентификатор разделяемого сегмента

void \*shared\_mem; // адрес привязки к процессу

FILE \*fpOut; // файл для записи данных

int pid =getpid();

//создание разделяемого сегмента

shm\_id = shmget(128, 4096, 0606 | IPC\_CREAT | IPC\_EXCL);

if(shm\_id != -1){

is\_creator = true;

puts("Разделяемый сегмент создан");

}else{

shm\_id = shmget(128, 4096, IPC\_CREAT);

if(shm\_id == -1){

puts("Не удается подключиться к разделяемому сегменту");

return 1;

}else{

puts("Подключен к разделяемому сегменту");

}

}

// присоединение сегмента к процессу

shared\_mem = shmat(shm\_id, 0, 0);

if (is\_creator) {

for (int i = 0; i < n; i++) \*(bool\*)(shared\_mem + i) = false;

for (int i = 0; i < n; i++) \*(int\*)(shared\_mem + n + 4\*i) = 0;

}

// алгоритм Лампорта

for(int i=0;i < \*argv[2] - '0';i++) {

\*(bool\*)(shared\_mem + prog\_id) = true;

memcpy(number,shared\_mem + n, 4\*n);

\*(int\*)(shared\_mem + n + 4\*prog\_id) = max\_num(number) + 1;

\*(bool\*)(shared\_mem + prog\_id) = false;

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

for(int j = 0; j < n; j++) {

while(\*(bool\*)(shared\_mem + j) ==1);

while(wait\_other(\*(int\*)(shared\_mem + n + 4\*j),\*(int\*)(shared\_mem + n + 4\*j),j,\*(int\*)(shared\_mem + n + 4\*prog\_id),prog\_id));

}

// начало критической секции

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

printf("Программа %d, PID %d, номер %d\n", prog\_id,pid, \*(int\*)(shared\_mem + n + 4\*prog\_id));

fprintf(fpOut, "Программа %d, PID %d, номер %d\n", prog\_id, pid, \*(int\*)(shared\_mem + n + 4\*prog\_id));

fclose(fpOut);

// конец критической секции

\*(int\*)(shared\_mem + n + 4\*prog\_id) = 0;

}

// отсоединение сегмента

memcpy(number,shared\_mem + n, 4\*n);

shmdt(shared\_mem);

bool f = true;

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

if(number[i] != 0)

f=false;

// удаление разделяемого сегмента

if (f) {

if (shmctl(shm\_id, IPC\_RMID, 0) == 0) puts("Разделяемый сегмент удалён");

}

return 0;

}

Вывод

В процессе выполнения лабораторной работы познакомились с организацией разделяемой памяти и системными функциями, обеспечивающими обмен данными между процессами.