**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

Государственное образовательное учреждение

высшего профессионального образования

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**СВЯЗИ И ИНФОРМАТИКИ**

Кафедра Математической кибернетики и информационных технологий

Лабораторная работа №4

на тему «Знакомство с процессами, передачей данных

между процессами и их синхронизацией»

по дисциплине «Современные операционные системы»

Выполнила: студ. гр. ЗМПП1901

Иванова Е.В.

Проверила: И.о. зам. зав. кафедрой Мосева М.С.

Москва 2019

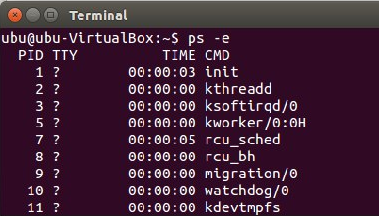
**Цель работы:** практическое знакомство с объектом «процесс», основными механизмами передачи данных между процессами, а также синхронизацией взаимодействующих процессов в ОС Unix.

**Задание:** Изучить базовые возможностей оболочки bash ОС Unix по управлению процессами (заданиями). Разработать приложения, реализующие схему «клиент–сервер» с использованием средств межпроцессного взаимодействия: семафоров, разделяемой памяти, программных каналов и одной очереди сообщений.

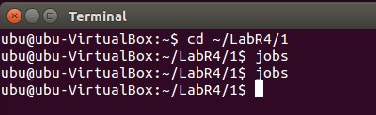
**Выполнение:**

**1. Запустить несколько заданий (например, команд просмотра файлов less), возвращаясь в командную строку комбинацией клавиш Ctrl-Z, и изучить действие команд ps, jobs, fg, bg, kill, killall.**

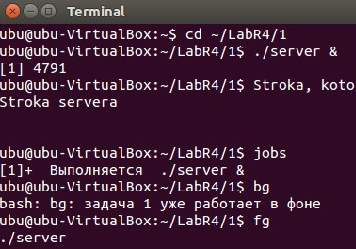
Команда **ps** выдает список протекающих в настоящее время процессов.



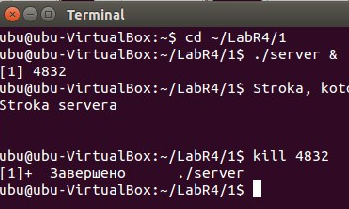
Команда **jobs** выдает список процессов, запущенных пользователем.

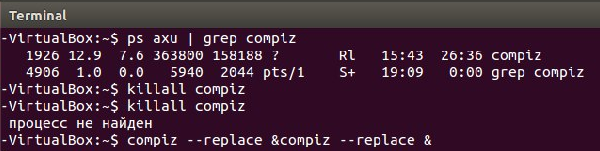


Команды **fg** и **bg** служат для возобновления приостановленного процесса соответственно на «переднем плане» или в фоновом режиме.



Команда **kill** служит для уничтожения процессов, а **killall** – для завершения процессов по имени.

****

****

**2.** **Обеспечить синхронизацию процессов и передачу данных между ними на примере приложений «клиент» и «сервер», создав два процесса (исполняемых файла): «клиент» (первый исполняемый файл) и «сервер» (второй исполняемый файл). С помощью механизмов межпроцессного взаимодействия обеспечить передачу информации от «клиента» к «серверу» и наоборот. В качестве типа передаваемой информации можно использовать: данные, вводимые с клавиатуры; данные, считываемые из файла; данные, генерируемые случайным образом и т. п.**

**Обмен данными между процессами «клиент–сервер» осуществить следующим образом:**

* **с использованием программных каналов (именованных либо неименованных, по указанию преподавателя);**
* **с использованием (по указанию преподавателя) одного из перечисленных вариантов:**
* **разделяемая память (обязательна синхронизация процессов, например, с помощью семафоров);**
* **очередь сообщений.**

**2.1. Обмен данными между процессами «клиент–сервер» с использованием именованных программных каналов.**

Есть 2 процесса (исполняемых файлов) «клиент» и «сервер». Создаются 2 именованных канала один для передачи сообщения от клиента к серверу, другой – от сервера к клиенту.

Программа-сервер:

#include <unistd.h>

#include <stdio.h>

#include <error.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <iostream>

#include <cstring>

#include <fstream>

#include <sys/stat.h>

#include <errno.h>

#include <fcntl.h>

#include <stdlib.h>

using namespace std;

#define MAXLINE 128

#define FILE\_MODE (S\_IRUSR|S\_IWUSR|S\_IRGRP|S\_IROTH)

#define FIFO1 "fifo.1"

#define FIFO2 "fifo.2"

int main(){

int read1 = -1, write1 = -1;

size\_t n=0;

char stroka1[MAXLINE];

cout<<("Stroka, kotoryu dobavlyaet server")<<endl;

cout<<strcpy(stroka1,"Stroka servera ")<<endl<<endl;

write1 = open(FIFO1, O\_WRONLY,0);

write(write1,stroka1,strlen(stroka1));

read1 = open(FIFO2, O\_RDONLY,0);

if ((read1!=-1)){

while ((n = read(read1,stroka1,MAXLINE))>0) {

stroka1[n] =0;

break;

}}

cout<<("Rezultat na servere")<<endl;

cout<<stroka1<<endl;

close(read1);

close(write1);

return 1;

}

Программа-клиент:

#include <unistd.h>

#include <stdio.h>

#include <error.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <iostream>

#include <cstring>

#include <fstream>

#include <sys/stat.h>

#include <errno.h>

#include <fcntl.h>

#include <stdlib.h>

using namespace std;

#define MAXLINE 128

#define FILE\_MODE (S\_IRUSR|S\_IWUSR|S\_IRGRP|S\_IROTH)

#define FIFO1 "fifo.1"

#define FIFO2 "fifo.2"

int main(){

int read1 = -1, write1 = -1;

ssize\_t n;

char stroka1[MAXLINE];

cout<<("Stroka dlya otpravki servery")<<endl;

cout<<"stroka clienta"<<endl<<endl;

unlink(FIFO1);

unlink(FIFO2);

if (mkfifo(FIFO1, FILE\_MODE)==EEXIST)cout<<"\n Pipes is exist"<<endl;

if (mkfifo(FIFO2, FILE\_MODE)==EEXIST)cout<<"\n Pipes is exist"<<endl;

read1 = open(FIFO1, O\_RDONLY);

while((n=read(read1,stroka1,MAXLINE))>0){

stroka1[n]=0;

break;}

strcat(stroka1, " stroka clienta\n ");

write1 = open(FIFO2, O\_WRONLY);

if ((write1 != -1)) write(write1,stroka1,strlen(stroka1));

cout<<("Rezultat s servera")<<endl;

cout<<stroka1<<endl;

close(read1);

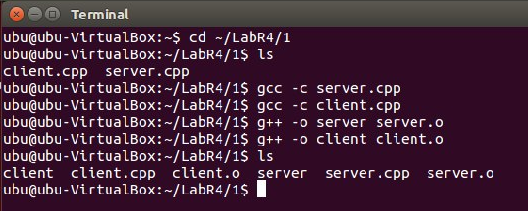
close(write1);

unlink(FIFO1);

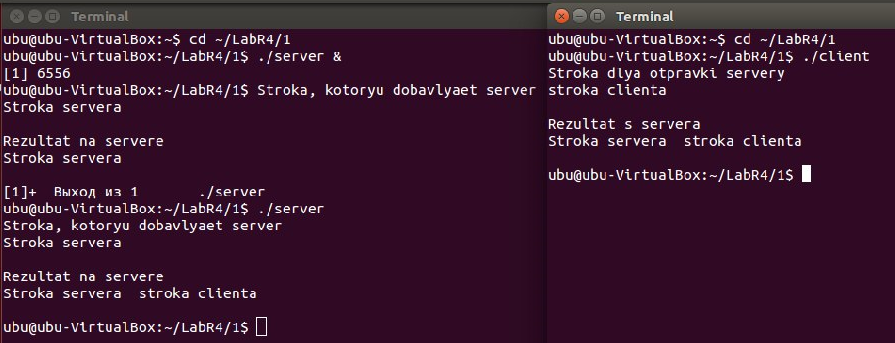
unlink(FIFO2);

}

Результат:



«Клиент» отправляет строку «серверу», «сервер» к этой строке добавляет свою и отправляет обратно клиенту (сначала запускается серверный процесс в фоновом режиме, а затем клиентский).



**2.2. Обмен данными между процессами «клиент–сервер» с использованием очереди сообщений.**

Программа-сервер записывает в очередь случайные числа, программа-клиент считывает из очереди эти числа, выводит их на экран, извлекает корень из суммы этих чисел, выводит результат и удаляет очередь.

Программа-сервер:

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <fcntl.h>

#include <error.h>

#include <iostream>

#include <stdlib.h>

#define MAXLINE 128

using namespace std;

#define MYKEY 4321L

#define FLAGS (S\_IRUSR|S\_IWUSR|S\_IRGRP|S\_IROTH)

struct msg{

long mt;

int n;

};

int main (){

printf("Ochered sozdana\n");

int msgid = msgget(ftok("ochered",0), FLAGS|IPC\_CREAT);

cout<<msgid<<endl;

struct msg s\_msg;

s\_msg.mt =1;

printf("Proischodit otpravka dannih\n");

for (int i=0;i<7; i++){

s\_msg.n =rand()%50;

msgsnd(msgid, &s\_msg, sizeof(s\_msg), IPC\_NOWAIT);

}

printf("Zavershenie raboti servera");

cout<<endl;

return 0;

}

Программа-клиент:

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <fcntl.h>

#include <error.h>

#include <iostream>

#include <stdlib.h>

#include <cmath>

using namespace std;

#define MYKEY 4321L

#define FLAGS (S\_IRUSR|S\_IWUSR|S\_IRGRP|S\_IROTH)

struct msg{

long mt;

int n;

};

int main()

{

printf("Ochered otkrita\n");

int msgid = msgget(ftok("ochered",0), FLAGS);

cout<<msgid<<endl;{

struct msg c\_msg;

c\_msg.mt =1;

int count = 0, s = 0, rez = 0;

while (msgrcv(msgid, &c\_msg,sizeof(c\_msg), 0, IPC\_NOWAIT)>0){

count++;

s+=c\_msg.n;

printf("Chislo: %d\n", c\_msg.n);

};

rez = round(sqrt(s));

cout<<endl;

printf("Rezultat: %d\n", rez);

printf("Ochered ydalena \n");

cout<<endl;

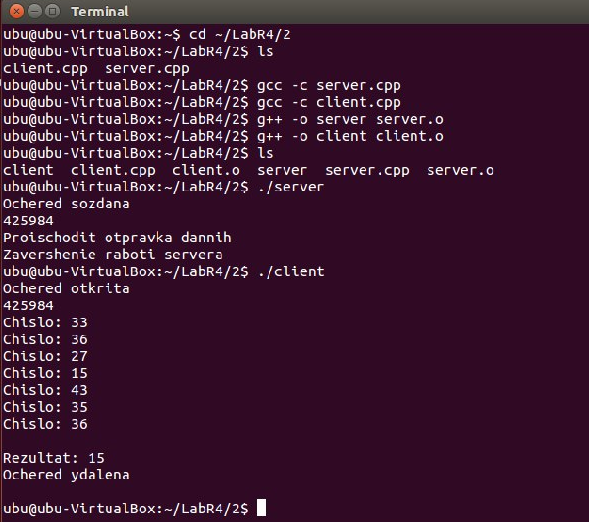
msgctl(msgid, IPC\_RMID, NULL);

}

return 0;

}

Результат:

****