**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ   
УНИВЕРСИТЕТ им. А. И. Герцена»**

ИНСТИТУТ КОМПЬЮТЕНЫХ НАУК И ТЕХНОЛОГИЧЕСКОГО ОБРАЗОВАНИЯ

**КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ЭЛЕКТРОННОГО ОБУЧЕНИЯ**

**ОТЧЕТ**

**По лабораторной работе №4**

**«ПРАКТИЧЕСКОЕ ЗНАКОМСТВО С ПРОЦЕССАМИ, ПЕРЕДАЧЕЙ ДАННЫХ МЕЖДУ ПРОЦЕССАМИ И ИХ СИНХРОНИЗАЦИЕЙ»**

**по курсу «Операционные системы»**

**Студенки 2 курса**

**Волковой А. И.**

**Преподаватель:**

**профессор кафедры**

**компьютерных**

**технологий и**

**электронного**

**обучения**

**Абрамян Г. В.**

**Санкт-Петербург**

**2017 г.**

## 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 работы**

**Задание 1**

Программа-клиент записывает в файлы слова, которые посылает программа-сервер. По слову exit обе программы прекращают работать. Листинг на основе очереди FIFO.

**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>

#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 argc, char \*\*argv)

{

int readfd =-1, writefd = -1;

size\_t n = 0;

char str[MAXLINE];

char str2[MAXLINE];

cout<<"Server start..."<

unlink(FIFO1);

unlink(FIFO2);

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

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

cout<<"You may type words..."<

writefd = open(FIFO2, O\_WRONLY);

if ((writefd!=-1)){

while (1){

cin>>str;

write(writefd, str, strlen(str));

readfd = open(FIFO1, O\_RDONLY);

while ((n = read(readfd, str2, MAXLINE))>0) {

str2[n] =0;

cout<

break;

}

if (strcmp(str, "exit")==0) break;

}

close(readfd);

close(writefd);

unlink(FIFO1);

unlink(FIFO2);

cout<<"Server is shut down"<

}else cout<<"can't open pipes.."<

return 0; }

**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>

#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 argc, char \*\*argv)

{

int readfd =-1, writefd = -1;

size\_t n = 0;

char str[MAXLINE];

char str2[MAXLINE];

ofstream cout("log.txt");

cout<<"Client was running..."<

while (1)

{

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

if (readfd!=-1)

{

while ((n=read(readfd, str, MAXLINE))>0)

{

str[n] = 0;

cout<

break;

}

strcpy(str2, "ok from client");

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

write(writefd, str2, strlen(str2));

close(readfd);

close(writefd);

}

sleep(1);

if (strcmp(str, "exit") ==0) break;

}

cout<<"Client is shut down";

cout.close();

return 0;

}

**Результаты работы сервера**

Server start...

You may type words...

one

ok from client

two

ok from client

three

ok from client

exit

ok from client

Server is shut down

**Результат работы клиента**

Client was running..

one

two

three

exit

Client is shut down

**Задание 2**

Программа-сервер записывает в очередь числа, программа-клиент считывает из очереди эти числа, считает их количество и сумму, удаляет очередь.

**Сервер**

#include <stdio.h>

#include <error.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <iostream>

# #include <fcntl.h>

#include <stdlib.h>

#define MYKEY 4321L

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

struct msg

{

long mtype;

int number;

};

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

{

printf("Server :Make message queue\n");

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

cout<< msgid<

struct msg s\_msg;

s\_msg.mtype =1;

printf("Server :Sending messages\n");

for (int i=1; i<100; i++)

{

s\_msg.number = i;

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

}

printf("Server :Log out");

return 0;

}

**Клиент**

#include <stdio.h>

#include <error.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <iostream>

# #include <fcntl.h>

#include <stdlib.h>

#define MYKEY 4321L

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

struct msg

{

long mtype;

int number;

};

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

{

printf("Client :Open message queue\n");

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

cout<

// if (msgid> 0)

{

struct msg c\_msg;

c\_msg.mtype =1;

printf("Client :Receiving messages\n");

int count=0, sum =0;

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

{

count++;

sum+=c\_msg.number;

};

printf("Client :The count is %d\n", count);

printf("Client :The sum is %d\n", sum);

printf("Client :Removing queue \n");

msgctl( msgid, IPC\_RMID, NULL);

}

printf("Server :Log out");

return 0;

}

**Результаты работы сервера**

Server :Make message queue

0

Server :Sending messages

Server :Log out

**Результаты работы клиента**

Client :Open message queue

0

Client :Receiving messages

Client :The count is 99

Client :The sum is 198

Client :Removing queue

**Выводы:**

В ходе проделанной лабораторной работы мы познакомились с основными механизмами передачи данных между процессами, а также их синхронизацией. Синхронизация процессов мало отличается от синхронизации потоков, поэтому это очень важное и кропотливое дело.