Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет» Кафедра ИИТ

Лабораторная работа №6

по дисциплине «Операционные системы и системное программирование»

Тема: «Средства межпроцессного взаимодействия»

Вариант 7

Выполнил: студент 2-го курса группы ПО-6 Лавренчик Д.О.

Проверил: Давидюк Ю.И.

Брест 2022

**Лабораторная работа №6**

**Ход работы**

***Задание для выполнения:***

Ознакомиться с руководством, теоретическими сведениями и лекционным

материалом по использованию и функционированию средств взаимодействия.

Написать программу, которая порождает дочерний процесс, и общается с ним через

средства взаимодействия согласно варианту (табл.А), передавая и получая информацию

согласно варианту (табл.Б). Передачу и получение информации каждым из процессов

сопровождать выводом на экран информации типа "процесс такой-то передал/получил

такую-то информацию". Дочерние процессы начинают операции после получения

сигнала SIGUSR1 от родительского процесса.



**Код программы:**

#define SIZE 3

#include <fcntl.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <signal.h>

#include <assert.h>

#include <pthread.h>

#include <time.h>

void hdl(int sig)

{

if(sig == SIGUSR1)

printf("SIGUSR1...\n");

else if(sig == SIGUSR2)

printf("SIGUSR2...\n");

else

printf("Something else\n");

}

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

srand(time(0));

struct sigaction act;

memset(&act, 0, sizeof(act));

act.sa\_handler = hdl;

sigset\_t set;

sigemptyset(&set);

sigaddset(&set, SIGUSR1);

sigaddset(&set, SIGUSR2);

act.sa\_mask = set;

sigaction(SIGUSR1, &act, 0);

sigaction(SIGUSR2, &act, 0);

signal(SIGUSR1, hdl);

int sig=0,fd;

pid\_t pid;

pid=fork();

if (pid > 0) {

printf("<Parent>\n");

printf("PID = %d\n", getpid());

off\_t length=SIZE\*sizeof(int);

//получаем доступ к памяти и исходные данные

fd = shm\_open("mem", O\_RDWR | O\_CREAT, 0666);

ftruncate(fd,length);

assert(fd > 0);

int \*ptr = (int\*)mmap(NULL,length, PROT\_READ | PROT\_WRITE,MAP\_SHARED, fd, 0);

assert(ptr);

for (size\_t i = 0; i < SIZE; i++) {

ptr[i] = 1+rand() % 10;

}

printf("a = %d \n",ptr[0]);

printf("b = %d \n",ptr[1]);

printf("c = %d \n",ptr[2]);

//посылаем сигнал дочернему процессу

kill(pid, SIGUSR1);

sigemptyset(&set);

sigaddset(&set, SIGUSR2);

printf("Parent wait signal...\n");

//ожидаем сигнал SIGUSR1

sigwait(&set, &sig);

printf("Parent gets perimeter P = %d\n", ptr[0]);

printf("Parent exit\n");

close(fd);

}

else if (pid == 0) {

sigemptyset(&set);

sigaddset(&set, SIGUSR1);

//ожидаем сигнал SIGUSR1

sigwait(&set, &sig);

printf("<Child>\n");

printf("PID = %d\n", getpid());

printf("work... \n");

fd = shm\_open("mem", O\_RDWR, 0666);

assert(fd > 0);

struct stat sb;

fstat(fd, &sb);

off\_t sizem = sb.st\_size;

int\* ptr = (int\*)mmap(NULL, sizem, PROT\_READ | PROT\_WRITE,MAP\_SHARED, fd, 0);

//принимаем исходные данные и обрабатываем

assert(ptr);

int p = 0;

for (size\_t i = 0; i < SIZE; i++) {

p = p + ptr[i];

}

ptr[0] = p;

sleep(2);

//посылаем сигнал родительскому процессу

kill(getppid(), SIGUSR2);

printf("Child exit\n");

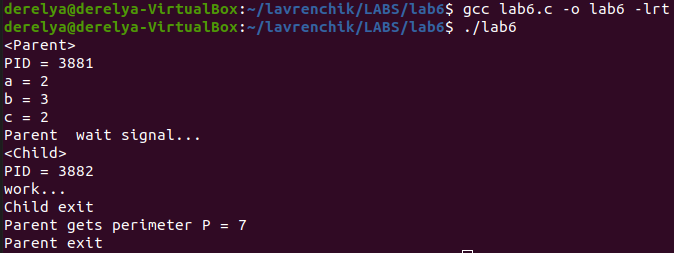
exit(0);

}

return 0;

}

**Результат выполнения программы:**

****

**Вывод:** в ходе выполнения данной лабораторной работы ознакомился с руководством, теоретическими сведениями и лекционным материалом по использованию и функционированию средств взаимодействия. Написал программу, которая порождает дочерний процесс, и общается с ним через

средства взаимодействия (разделяемая память)