Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа

Дисциплина: «Операционные системы»

III семестр

Задание 2: «Управление процессами в ОС»

|  |  |
| --- | --- |
| Группа: | М8О-108Б-18, №6 |
| Студент: | Васильева Василиса Евгеньевна |
| Преподаватель: | Миронов Евгений Сергеевич |
| Оценка: |  |
| Дата: | 29.11.2019 |

Москва, 2019

1. **Задание**

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решения задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe).

Вариант 14: Родительский процесс считывает числа со стандартного потока ввода. Дочерний процесс вычисляет квадратный корень этих чисел и передает результат на печать родительскому процессу.

1. **Адрес репозитория на GitHub**

https://github.com/vasilisavasileva/OS\_1

1. **Код программы на С**

sever.c

#include<stdio.h>

#include<math.h>

#include<Windows.h>

#include<tchar.h>

int \_tmain(int argc, \_TCHAR\* argv[]) {

HANDLE inH = GetStdHandle(STD\_INPUT\_HANDLE);

HANDLE outH = GetStdHandle(STD\_OUTPUT\_HANDLE);

double l;

DWORD ReadBytes, WriteBytes;

BOOL isSuccess;

int n;

isSuccess = ReadFile(inH, &n, sizeof(double), &ReadBytes, NULL);

if (!isSuccess)

exit(1);

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

ReadFile(inH, &l, sizeof(double), &ReadBytes, NULL);

l = sqrt(l);

WriteFile(outH, &l, sizeof(double), &WriteBytes, NULL);

}

CloseHandle(inH);

CloseHandle(outH);

return 0;

}

*client.c*

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<Windows.h>

#include<tchar.h>

#include<stdbool.h>

int \_tmain(int argc, \_TCHAR\* argv[]) {

SECURITY\_ATTRIBUTES sa;

sa.nLength = sizeof(SECURITY\_ATTRIBUTES);

sa.lpSecurityDescriptor = NULL;

sa.bInheritHandle = TRUE;

TCHAR SecondProcess[] = "server";

HANDLE pipe1Read, pipe1Write, pipe2Read, pipe2Write;

CreatePipe(&pipe1Read, &pipe1Write, &sa, 0);

CreatePipe(&pipe2Read, &pipe2Write, &sa, 0);

PROCESS\_INFORMATION ProcessInfo;

ZeroMemory(&ProcessInfo, sizeof(PROCESS\_INFORMATION));

STARTUPINFO StartupInfo;

ZeroMemory(&StartupInfo, sizeof(STARTUPINFO));

StartupInfo.cb = sizeof(STARTUPINFO);

StartupInfo.dwFlags = STARTF\_USESTDHANDLES;

StartupInfo.hStdInput = pipe1Read;

StartupInfo.hStdOutput = pipe2Write;

BOOL process = CreateProcess(NULL,

SecondProcess,

NULL, NULL, TRUE,

CREATE\_NEW\_CONSOLE,

NULL, NULL,

&StartupInfo,

&ProcessInfo);

CloseHandle(pipe1Read);

CloseHandle(pipe2Write);

DWORD WriteBytes, ReadBytes;

double l;

BOOL isSuccess;

if (process == 1)

printf("process true\n");

else {

printf("error\n");

exit(1);

}

printf("Enter the count of number: ");

int n;

scanf("%d", &n);

isSuccess = WriteFile(pipe1Write, &n, sizeof(int), &WriteBytes, NULL);

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

scanf("%lf", &l);

isSuccess = WriteFile(pipe1Write, &l, sizeof(double), &WriteBytes, NULL);

isSuccess ? printf("Sended\n") : printf("Not send\n");

isSuccess = ReadFile(pipe2Read, &l, sizeof(double), &ReadBytes, NULL);

isSuccess ? printf("Received\n") : printf("Not received\n");

printf("res: %f\n", l);

}

CloseHandle(pipe1Write);

CloseHandle(pipe2Read);

CloseHandle(ProcessInfo.hThread);

CloseHandle(ProcessInfo.hProcess);

system("pause");

return 0;

}

1. **Результаты выполнения тестов**

|  |  |
| --- | --- |
| Входные данные | Результат |
| 4 | res: 2.000000 |
| 90 | res: 9.486833 |
| 25 | res: 5.000000 |

1. **Объяснение результатов работы программы**

Работа программы разделена на два файла: “client.c” и “server.c”. В «клиенте» происходит определение атрибутов безопасности, дескрипторов на ввод и вывод, структура SturtupInfo, создаются каналы для передачи данных между потоками (pipes). После ввода количества чисел, которые будут поданы в обработку, программа начинает их принимать. На каждом шаге мы проверяем, корректно ли идет работа, и периодически выводим сообщения на экран. Так мы знаем, что ввод прошел успешно, что данные отправлены в дочерний процесс и возвращены обратно. После того, как мы убедились, что введенное число было отправлено на обработку в «сервер», мы приступаем к работе уже в «сервере». Принимаем входное число в дочернем процессе, извлекаем квадратный корень и отправляем назад в «клиент». Клиент результат принимает и распечатывает. Каналы закрываются. Работа программы завершается.

1. **Вывод**

При введении дочерних процессов мы возлагаем на них часть работы, которая должна быть выполнена в ходе программы. В нашем случае, мы отдаем дочернему процессу извлечение квадратного корня из входных чисел. Это позволяет упростить основной код и разбить его на задачи.