Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное   
образовательное учреждение высшего образования   
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе №3

Дисциплина: «Операционные системы»

Тема «Межпроцессное взаимодействие в операционной  
системе Ubuntu»

Выполнила: Иванова А.А.

Группа: 6206-010302D

Самара, 2024

**Задание на лабораторную работу**

Необходимо написать программу, которая разделяется на две части. Первая часть – клиентская – взаимодействует с пользователем, содержит интерфейс ввода вывода чисел из консольного ввода. Желательно обойтись без передачи параметров расчета через параметры программы (argv). Справка так же выводится по ключу –help, как и ранее. Вторая часть (серверная) ничего не выводит на экран, только принимает информацию от клиентской части, производит вычисления в зависимости от варианта ниже и отправляет клиентской части результат.

Программа должна разделяться на две части при помощи системного вызова fork(). Создание неименованных каналов должно происходить до вызова fork().

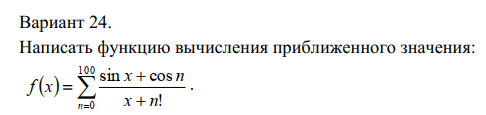
Исходный код и клиентской, и серверной частей программы можно разместить в одном файле внутри одной функции main. При отладке программы в отладчике gdb необходимо помнить о том, что клиентская и серверная части работают одновременно и представляют собой разные процессы.

Другой вариант написания лабораторной работы – подмена стандартных ввода и вывода (функция int dup2(int oldfd, int newfd)), и запуск расчетной и клиентской частей через функцию семейства exec. При этом oldfd будет указывать на ваш созданный канал, newfd – на файл, соответствующий стандартному потоку ввода вывода – 0 или 1. В этом случае лабораторная работа будет состоять из трех .cpp файлов, и трех запускных, соответственно.

Для корректной работы необходимо создать два канала, каждый из которых описывается массивом двух чисел. Один канал будет использоваться для передачи данных и команд серверу от клиента, второй – для передачи результата от клиента к серверу.

Таким образом, стандартный набор действий следующий:  
1. Проверка командной строки на ключ --help. При наличии такового должен быть выведен help и программа должна завершить свою работу.  
2. Создание каналов.  
3. Разделение на клиентскую и серверную части.  
4. Работа  
5. После обмена (достаточно единственного) и сервер, и клиент могут завершить свою работу, команд для завершения работы сервера от клиента предусматривать не требуется.

Не забывайте закрывать каналы после использования.  
**Работа клиентской части:**1. Предложение ввода параметров, ожидание ввода с клавиатуры.  
2. Передача данных в канал на сервер  
3. Чтение результатов из канала от сервера  
4. Вывод результатов расчета на экран  
5. Выход.  
**Работа серверной части:**1. Чтение данных из канала от клиента.  
2. Расчет.  
3. Запись данных в канал клиенту.  
4. Выход



**Текст программы**

#include <iostream>

#include <fstream>

#include <cstring>

#include <string>

#include <math.h>

#include <unistd.h>

#include <sys/types.h>

using namespace std;

const int n = 100;

int check() {

cout << "Input x value: ";

int xx;

cin >> xx;

while (cin.fail() || cin.peek() != '\n') {

cout << "Error. Please, try again.\n";

cin.clear();

cin.ignore(10000, '\n');

cout << "Input x value: ";

cin >> xx;

}

return xx;

}

void Error1(string help) {

cout << "Error\n";

exit(1);

}

void Error2() {

cout << "Error\n";

exit(1);

}

void Help(){

cout << "Чтобы посчитать приближенное значение, запустите программу без ключей\n";

exit(0);

}

int factorial(int f){

int ff = 1;

for (int i = 1; i <= f; ++i) ff\*=i;

return ff;

}

double Result(double x){

double b = 0;

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

b += ((sin(x) + cos(i))/(x + factorial(i)));

}

return b;

}

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

if (argc == 2 && !strcmp(argv[1], "--help")) Help();

else if (argc != 1) Error1("Error! Run the program with the '--help' key");

else{

cout << "0 - Данные вводит сам пользователь\n";

cout << "1 - Данные считываются из файла\n";

cout << "Вариант 24. Написать функцию вычисления приближенного значения.\n";

cout << "f(x) = SUM(0\_n)(sin(x)+cos(n))/(x+n!)\n";

int x;

cout << "Выберите способ: ";

cin >> x;

if (x == 1){

cout << "Введите файл, из которого считать данные, и файл, в который записать результат" << endl;

string file1, file2;

cin >> file1 >> file2;

ifstream fin;

fin.open(file1, ios::binary);

if (fin.is\_open()) {

double data;

fin >> data;

if (fin.fail() || fin.peek() != ' ') {cout <<"Error!"; exit(1); }

double res = Result(data);

ofstream fout;

fout.open(file2, ios::binary);

fout << res;

fin.close();

fout.close();

}

else Error2();

}

else if (x == 0) {

//Неименованные каналы связи in и out

int pipe\_in[2];

int pipe\_out[2];

pipe(pipe\_in);

pipe(pipe\_out);

pid\_t pid; //хранение идентификатора процесса

pid = fork();

if (pid > 0)

{

double x = check();

write(pipe\_in[1], &x, sizeof(double));

double a;

read(pipe\_out[0], &a, sizeof(double));

cout << "Result: " << a << endl;

}

else{

double x;

read(pipe\_in[0], &x, sizeof(double));

double result = Result(x);

write(pipe\_out[1], &result, sizeof(double));

}

for (int i=0; i<2; ++i){

close(pipe\_in[i]);

close(pipe\_out[i]);

}

}

}

return 0;

}