Министерство образования и науки Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра теоретической и прикладной информатики

# Лабораторная работа №2

## по дисциплине «Управление ресурсами в вычислительных системах»

Факультет: ПМИ

Группа: ПМИ-12

Вариант: 10

Студенты: Курдюков И.Н.

Омельницкая Е.И.

Преподаватели: Стасышин В.М., Сивак М.А.

Новосибирск

2024

# Цель работы

Изучить программные средства создания процессов, получить навыки управления и синхронизации процессов, а также простейшие способы обмена данными между

процессами. Ознакомиться со средствами динамического запуска программ в рамках порожденного процесса, изучить механизм сигналов ОС UNIX, позволяющий

процессам реагировать на различные события, и каналы, как одно из средств обмена информацией между процессами.

# Задание

Разработать программу, вычисляющую интеграл на отрезке [A;B] от функции f(x)=(exp(x)-exp(-x))/2 методом трапеций, разбивая интервал на K равных отрезков. Для нахождения exp(х) и exp(-x) программа должна породить параллельные процессы, вычисляющие эти значение путём разложения в ряд по формулам вычислительной математики.

# Описание используемых структур

### int fork()

Порождение процесса-потомка, точной копии процесса-предка. Процесс-потомок в качестве возвращаемого значения системного вызова fork() получает 0, а

процесс-предок - идентификатор процесса-потомка.

### void exit(int value)

Выполняет немедленное завершение программы. Аргумент параметра value возвращается принимающей стороной (ОС или другой программой) в родительский процесс. Как правило, возвращается значение 0 или EXIT\_SUCCESS указывает на успешное завершение программы, и любое другое значение или значение макроса EXIT\_FAILURE используется для указания об аварийном завершении программы.

### int wait(int \*status)

В результате осуществления процессом системного вызова wait() функционирование процесса приостанавливается до момента завершения порожденного им процесса-потомка. По завершении процесса-потомка процесс предок пробуждается и в качестве возвращаемого значения системного вызова wait() получает идентификатор завершившегося процесса-потомка.

# Спецификация

Программа разработана и протестирована на компьютере с Cygwin. Cygwin — UNIX-подобная среда и интерфейс командной строки для Microsoft Windows. В качестве компилятора используется GCC версии 11.2.0.

Директория, содержащая файл: C:\cygwin64\home\Ivan

Название файла с программой: upres2lab\_6**.c**

Название файла для вывода: **output.txt**

Компиляция программы: **gcc -o <результат компиляции> upres2lab\_6.c**

Запуск программы: **./<результат компиляции>**

# Описание алгоритма

1. Начинаем расчет методом трапеций, на этапе вычисления значения функции в точке вызываем функцию f(x).

2. В данной функции подсчитывается значение функции в точке х используя разложение exp(x) и exp(-x) в ряд тейлора.

3. В результате выводим ответ в файл и на консоль.

# Код программы

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#define A 0.0

#define B 1.0

#define K 100 // Количество разбиений отрезка [A;B]

double calculateExp(double x) { //вычисляем exp(x) путем разложения в ряд тейлора до 51 члена

    double result = 1;

    double term = 1;

    double numerator = 1;

    double denominator = 1;

    for (int i = 1; i <= 51; i++) {

        numerator \*= x;

        denominator \*= i;

        term = numerator / denominator;

        result += term;

    }

    return result;

}

double f(double x) // вычисляем заданную функцию

{

    int pid1 = fork();

    if (pid1 == 0)

    {

        // Дочерний процесс 1 для exp(x)

        double result1 = calculateExp(x);

        FILE\* file = fopen("tempfile.txt", "a");

        fprintf(file, "1 %.10f\n", result1); // Записываем результат в файл

        fclose(file);

        exit(0);

    }

    else

    {

        int pid2 = fork();

        if (pid2 == 0)

         {

            // Дочерний процесс 2 для exp(-x)

            double result2 = calculateExp(-x);

            FILE\* file = fopen("tempfile.txt", "a");

            fprintf(file, "2 %.10f\n", result2); // Дописываем результат в файл

            fclose(file);

            exit(0);

        }

        else

        {

            // Родительский процесс

            wait(NULL); // Ждем, пока оба дочерних процесса завершатся

            wait(NULL);

        }

    FILE\* file = fopen("tempfile.txt", "r");

    if (file == NULL)

    {

        perror("Error opening file");

        return 1;

    }

    char buffer[255];

    double results[2];

    while (fscanf(file, "%s %lf", buffer, &results[atoi(buffer) - 1]) != EOF);

    fclose(file);

    return (results[1]-results[0]) / 2;

    }

}

int main()

{

    double h = (B - A) / K;

    double sum = 0.5 \* (f(A) + f(B)); // Начальное значение суммы

    for (int i = 1; i < K; i++)

    {

    double x = A + i \* h;

    sum += f(x);

    }

    double integral = h \* sum;

    printf("Calculated integral: %lf\n", integral);

    FILE\* file = fopen("output.txt", "w");

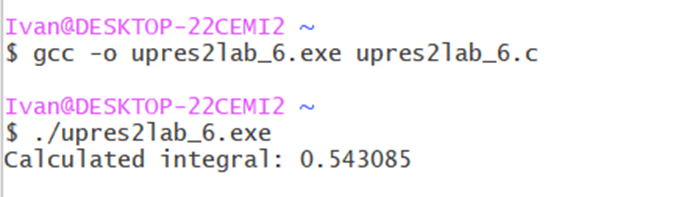
    fprintf(file, "Calculated integral: %lf\n", integral);

    return 0;

}

# Тестирование

Результат выполнения команды gcc -o upres2lab\_6.exe upres2lab\_6.c; ./upres2lab\_6.exe



Содержимое файла **output**:

