Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

Группа: М80-207Б-18

Студент: Петрин Сергей Александрович

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2019.

***Постановка задачи***Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке использовать стандартные средства создания потоков операционной системы (Windows/Unix).

*Вариант 9. Расчет n точек, заданных функцией. На вход подается количество точек и границы отрезка на котором производится расчет функции. Функция f(Xn) = (sin(Xn)+cos(Xn))\* f(Xn-1), f(X0) = sin(X0)+cos(X0). Необходимо, чтобы при выводе точки были выведены последовательно относительно x.*

***Алгоритм***

Напишем отдельную функцию, которая будет считать нам в заданной точке sin(x) + cos(x). С помощью потоков мы перенаправивим массив точек в нашу функцию и будем получать посчитанный массив точек. Дальше умножим точки предыдущую на текущую и произведём вывод.

***Описание программы***

Программа состоит из функции void \*threadFunc (void\* args), которая считает для точки функцию sin(x)+cos(x).

И главной функции int(main), которая считывает первую и вторую границы и количество точек. Разделив длину отрезка на количество точек, она получает массив точек и с помощью потока направляет в функцию void \*threadFunc (void\* args). Дальше мы каждую посчитанную точку умножаем на предыдущую и выводим это в ответ относительно Х.

***Листинг***

*#include <pthread.h>*

*#include <stdlib.h>*

*#include <mcheck.h>*

*#include <stdio.h>*

*#include <math.h>*

*void \*threadFunc(void\* args){*

*long double \*num = (long double\*) args;*

*\*num = sin(\*num) + cos(\*num);*

*return NULL;*

*}*

*int main() {*

*long double granica1;*

*long double granica2;*

*long double dlinaOtrezka;*

*int to4ki;*

*printf("Р’РІРµРґРёС‚Рµ РїРµСЂРІСѓСЋ РіСЂР°РЅРёС†Сѓ\n");*

*scanf("%Le", &granica1);*

*printf("Р’РІРµРґРёС‚Рµ РІС‚РѕСЂСѓСЋ РіСЂР°РЅРёС†Сѓ\n");*

*scanf("%Le", &granica2);*

*printf("Р’РІРµРґРёС‚Рµ РєРѕР»РёС‡РµСЃС‚РІРѕ С‚РѕС‡РµРє\n");*

*scanf("%d", &to4ki);*

*pthread\_t massivPotokov[to4ki - 1];*

*long double dots[to4ki - 1];*

*long double result[to4ki];*

*for(int i = 0; i < to4ki; i++) {*

*dlinaOtrezka = granica2 - granica1;*

*dots[i] = ( (dlinaOtrezka + 1) \* i / to4ki) + granica1;*

*}*

*for(int i = 0; i < to4ki; i++) {*

*pthread\_create(&massivPotokov[i], NULL, threadFunc, &dots[i]);*

*pthread\_join(massivPotokov[i], NULL);*

*}*

*printf("РўРѕС‡РєР° РҐ1 = %Lf\n", dots[0]);*

*for(int i = 1; i < to4ki; i++) {*

*dots[i] = dots[i] \* dots[i - 1];*

*printf("РўРѕС‡РєР° X%d = %Lf\n", i + 1, dots[i]);*

*}*

*}*

***Демонстрация работы программы***

~/Course2/OS/laba3(master\*) # gcc -std=c99 laba3.c -o a.out -lpthread -lm root@DESKTOP-DL6JCR0

---------------------------------------------------------------------------------------------------------

~/Course2/OS/laba3(master\*) # ./a.out root@DESKTOP-DL6JCR0

Введите первую границу

5

Введите вторую границу

10

Введите количество точек

5

Точка Х1 = -0.675262

Точка X2 = -0.616820

Точка X3 = -0.824846

Точка X4 = -0.045925

Точка X5 = 0.059560

---------------------------------------------------------------------------------------------------------

~/Course2/OS/laba3(master\*) # ./a.out root@DESKTOP-DL6JCR0

Введите первую границу

1

Введите вторую границу

10

Введите количество точек

5

Точка Х1 = 1.381773

Точка X2 = -1.172949

Точка X3 = 0.792048

Точка X4 = 1.117492

Точка X5 = -0.557642

***Вывод***

В результате выполнения данной лабораторной работы я получил первый опыт работы с потоками в -nix системе.

Потоки связаны с процессами, и они могут показаться похожими (и те и те являются независимыми единицами выполнения), однако они имеют существенные различия: каждый процесс выполняется в отдельном адресном пространстве, каждому процессу выделяются свои отдельные ресурсы, а взаимодействие процессов реализуется при помощи дополнительных инструментов (например, pipe), в то время как потоки используют общую область памяти, имеют собственные регистры и стек, доступ к которым могут получать другие потоки.

работа с потоками осуществляется при помощи системных вызовов библиотеки pthread.h:

pthread\_create – создание потока и выполнение необходимой функции (задается в аргументах к функции);

pthread\_exit – завершение работы потока;

pthread\_join – ожидание выполнения необходимого потока; при помощи этой функции можно синхронизировать потоки между собой (помогает решить проблему того, что потоки в программе часто могут выполняться не в очереди их создания).

Эти знания понадобятся мне при дальнейшей работе с ОС.