Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Операционные системы»

Управление потоками в ОС. Обеспечение синхронизации между потоками

Студент: П. Ф. Гришин
Преподаватель: Е. С. Миронов
Группа: М8О-201Б-21
Вариант: 4
Дата:
Оценка:
Подпись:

Москва, 2023

1 Постановка задачи

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Получившиеся результаты необходимо объяснить.

Отсортировать массив целых чисел при помощи битонической сортировки.

2 Постановка задачи

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки 2-мя способами:

- Во время компиляции (на этапе «линковки»/linking)
- Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

В лабораторной работе необходимо получить следующие части:

- Динамические библиотеки, реализующие контракты
- Тестовая программа (программа №1), которая используют одну из библиотек, используя знания полученные на этапе компиляции.
- Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты.

Провести анализ двух типов использования библиотек.

Пользовательский ввод должен быть организован следующим образом:

- Команда «0»: переключить одну реализацию контракты на другую
- Команда «1 args»: вызов первой функции контрактов
- Команда «2 args»: вызов второй функции контрактов

Контракты:

- Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e
- Рассчет значения числа Пи при заданной длине ряда (K)

3 Сведения о программе

Программа написанна на Си в Unix подобной операционной системе на базе ядра Linux.
Контракты описаны в файле functions.h, а реализация functions_1.c и functions_2.c.

1. Создание объектных файлов
2. Компиляция библиотек с ключем -shared. Получаем динамические библиотеки с расширением .so
3. Линковка библиотеки к необходимой программе

Для динамической загрузки библиотек используется библиотека dlfcn.h

4 Общий метод и алгоритм решения

Программа принимает в себя команды:

- В случае команды 1, мы считываем координаты левого и правого отрезка, интеграл функции которого хотим найти, а также шаг e, и находим интеграл либо методом прямоугольников, либо методом трапеций.
- В случае команды 2, мы считываем число - порядок ряда. Находим его либо с помощью ряда Лейбница, либо с помощью формулы Валлиса.
- В случае команды 0, мы закрываем старую библиотеку, открываем вторую и заменяем указатели на функции.

Для завершения программы нужно ввести комбинацию завершения ввода – CTRL+D.

5 Листинг программы

func.h

```
#ifndef FUNC_H
#define FUNC_H

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

double Integrate(double a, double b, double epsilon);
double Pi(int K);

#endif
```

lib1.c

```
double Integrate(double a, double b, double epsilon)
{
    int steps = fabs(b - a) / epsilon;
    double point = a;
    double result = 0;

    for (int i = 0; i < steps; ++i)
    {
        result += sin(point) * epsilon;
        point += epsilon;
    }

    return result;
}

double Pi(int K) {
    double pi = 0;
    for(int i = 0; i <= K; i++) {
        pi += 2.0/((4*i + 1)*(4*i + 3));
    }
    return pi * 4.0;
}
```

lib2.c

```
double Integrate(double a, double b, double epsilon)
{
    int steps = fabs(b - a) / epsilon;
    double point = a;
    double result = 0;

    for (int i = 0; i < steps; ++i)
    {
```

```
        result += sin(point + epsilon / 2) * epsilon;
        point += epsilon;
    }

    return result;
}

double Pi(int K) {
    double pi = 1;
    for (int i = 1; i <= K; i++) {
        pi *= (double)4*i*i/(4*i*i-1);
    }
    return (double)pi * 2;
}
```

main_dynamic.c

```
#include <stdlib.h>
#include <stdio.h>
#include <dlfcn.h>
#include <math.h>
#include <stdbool.h>

const char LIB1[] = "./libd1_dynamic.so";
const char LIB2[] = "./libd2_dynamic.so";

int main(int argc, char* argv[]) {
    void *library;
    bool flag = false;
    int x, K;
    double a, b, e;

    library = dlopen(LIB2, RTLD_LAZY);
    if (!library) {
        printf("Error dlopen(): %s\n", dlerror());
        return 1;
    }

    double(*Integrate)(double a, double b, double e);
    double(*Pi)(int K);
    *(void**)(&Integrate) = dlsym(library, "Integrate");
    *(void**)(&Pi) = dlsym(library, "Pi");

    for (;;) {
        scanf("%d", &x);
        if (x == 0) {
            dlclose(library);
            if (flag) {
                library = dlopen(LIB2, RTLD_LAZY);
                flag = false;
```

```
            } else {
                library = dlopen(LIB1, RTLD_LAZY);
                flag = true;
            }
            if (!library) {
                printf("Error dlopen(): %s\n", dlerror());
                return 1;
            }
            *(void**)(&Integrate) = dlsym(library, "Integrate");
            *(void**)(&Pi) = dlsym(library, "Pi");
        } else if (x == 1) {
            scanf("%lf %lf %lf", &a, &b, &e);
            printf("Result: ");
            double res = Integrate(a, b, e);
            printf("%lf\n", res);
        } else if (x == 2) {
            scanf("%d", &K);
            printf("Result: ");
            double res = Pi(K);
            printf("%lf\n", res);
        } else {
            dlclose(library);
            return 0;
        }
    }
    return 0;
}
```

main_static.c

```
#include "func.h"
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char* argv[]) {
    int x;
    double a, b, e;
    int K;

    for (;;) {
        scanf("%d", &x);
        if (x == 1) {
            scanf("%lf %lf %lf", &a, &b, &e);
            double res = Integrate(a, b, e);
            printf("%lf\n", res);
            x = 0;
        } else if (x == 2) {
            scanf("%d", &K);
            double res = Pi(K);
            printf("%lf\n", res);
```

```
            x = 0;
        } else {
            break;
        }
    }
    return 0;
}
```

CMakeListst.txt

```
add_library(d1_static STATIC src/lib1.c include/func.h)
add_library(d2_static STATIC src/lib2.c include/func.h)
add_library(d1_dynamic SHARED src/lib1.c include/func.h)
add_library(d2_dynamic SHARED src/lib2.c include/func.h)

add_executable(main_static_1 main_static.c)
add_executable(main_static_2 main_static.c)
add_executable(main_dynamic main_dynamic.c)

target_include_directories(main_static_1 PRIVATE include)
target_include_directories(main_static_2 PRIVATE include)

target_link_libraries(main_dynamic ${CMAKE_DL_LIBS})
target_link_libraries(main_static_1 PRIVATE d1_static)
target_link_libraries(main_static_2 PRIVATE d2_static)
target_include_directories(main_dynamic PRIVATE include)

find_library(MATH_LIBRARY m)

    target_link_libraries(d1_dynamic PUBLIC ${MATH_LIBRARY})
    target_link_libraries(d2_dynamic PUBLIC ${MATH_LIBRARY})
    target_link_libraries(main_static_1 PUBLIC ${MATH_LIBRARY})
    target_link_libraries(main_static_2 PUBLIC ${MATH_LIBRARY})
    target_link_libraries(main_dynamic PUBLIC ${MATH_LIBRARY})
```

6 Демонстрация работы программы

```
pavel@DESKTOP-K5KMLPV:~/Project/mai/2_course/OS/LB5$ gpavel@gpavel-HP-Pavilion-Gaming-
./main_static_1
1
0
3.14
0.1
1.995390
2
10
3.096162
^C
gpavel@gpavel-HP-Pavilion-Gaming-Laptop-17-cd1xxx:~/Desktop/OS/lab5$ ./main_static_2
1
0
3.14
0.1
1.999968
2
10
3.067704
^C
gpavel@gpavel-HP-Pavilion-Gaming-Laptop-17-cd1xxx:~/Desktop/OS/lab5$ ./main_dynamic
1
0
3.14
0.1
Result: 1.999968
2
10
Result: 3.067704
0
1
0
3.14
0.1
Result: 1.995390
2
10
Result: 3.096162
```

```
^C
gpavel@gpavel-HP-Pavilion-Gaming-Laptop-17-cd1xxx:~/Desktop/OS/lab5$
```

7 Вывод

Лабораторная работа была направлена на изучение динамических библиотек в Unix подобных операционных системах. Для изучения создания и работы с ними мною было написанно 2 программ: одна подлючала динамичкие библиотеки на этапе компиляции, а вторая во время исполнения.

Динамические библиотеки содержат функционал отдельно от программы и передают его непосредственно во время исполнения. Из плюсов такого подхода можно выделить, что во-первых, в таком случае размер результирующей программы меньше,во-вторых, одну и ту же библиотеку можно использовать в нескольких программах не встраивая в код, чем можно также добиться снижения общего занимаемого пространства на диске, и в-третьих, что после исправления ошибок в библиотеке не нужно перекомпилировать все программы, достаточно перекомпилировать саму библиотеку.

Однако у динамических библиотек есть и недостатки. Первый заключается в том, что вызов функции из динамической библиотеки происходит медленнее. Второй, что мы не можем подправить функционал библиотеки под конкретную программу не зацепив при этом других программ, работающих с этой библиотекой. И в-третьих, уже скомпилированная программа не будет работать на аналогичной системе без установленной динамической библиотеки.

Тем не менее плюсы динамичеких библиотек исчерпывают их минусы в большенстве задач, обратных случаях лучше обратиться к статическим библиотекам. В наше время с высокими мощностями вычислительных систем становится более важным сэкономить объем памяти, используемый программой, чем время обращения к функции. Поэтому динамические библиотеки используются в большенстве современных программ.