МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Институт №8 "Компьютерные науки и прикладная математика"
Кафедра 806 "Вычислительная математика и программирование"

Лабораторная работа №4

По курсу «Операционные системы»

Студент: Григорьев Т. А.
Группа: М8О-208Б-23
Преподаватель: Живалев Е. А.
Дата: _____
Оценка: _____
Подпись: _____

Москва, 2024

Тема: Динамические библиотеки

Цель работы: Приобретение практических навыков в:

- Создании динамических библиотек.
- Создании программ, использующих функции динамических библиотек.

Вариант: 29

Задачи:

1. Создать динамические библиотеки, реализующие два контракта:
    - Расчет числа π (Пи) двумя способами: по ряду Лейбница и по формуле Валлиса.
    - Перевод числа x из десятичной системы счисления: в двоичную, в троичную
2. Реализовать две программы для работы с библиотеками:
    - Первая программа использует функции библиотек на этапе компиляции.
    - Вторая программа загружает библиотеки динамически во время исполнения и позволяет переключаться между их реализациями.
3. Реализовать возможность переключения реализаций библиотек во время выполнения программы.

Ход работы:

1. Создание динамических библиотек Были реализованы две динамические библиотеки с различными реализациями функций для вычисления числа π и перевода в другую систему счисления.

Функции библиотеки №1:

- `float Pi(int K)`: Вычисление числа π по ряду Лейбница.
- `char* translation(long x)`: Перевод числа x из десятичной системы счисления в двоичную

Функции библиотеки №2:

- `float Pi(int K)`: Вычисление числа π по формуле Валлиса.
- `char* translation(long x)`: Перевод числа x из десятичной системы счисления в троичную

Репозиторий: https://github.com/timofeez/os-labs/tree/main/04

Код библиотеки №1:

```
#include <math.h>

float Pi(const int K) {
    float pi = 0.0;
    int sign = 1;
    for (int i = 0; i < K; i++) {
        pi += sign * 4.0 / (2 * i + 1);
        sign = -sign;
    }
    return pi;
}

char* translation(long x) {
    char* result = (char*)malloc(65);
    if (!result) return NULL;
    result[64] = '\0';
    int index = 63;
    do {
        result[index--] = '0' + (x % 2);
        x /= 2;
    } while (x > 0);
    return strdup(&result[index + 1]);
}
```

Код библиотеки №2:

```
float Pi(const int K) {
    float pi = 1.0;
    for (int i = 1; i <= K; i++) {
        pi *= (4.0 * i * i) / (4.0 * i * i - 1);
    }
    return pi * 2;
}
char* translation(long x) {
    char* result = (char*)malloc(65);
    if (!result) return NULL;
    result[64] = '\0';
    int index = 63;
    do {
```

```
        result[index--] = '0' + (x % 3);
        x /= 3;
    } while (x > 0);
    return strdup(&result[index + 1]);
}
```

2. Первая программа использует функции динамических библиотек на этапе компиляции. Пользователь может вызывать функции для вычисления числа π и числа x, вводя соответствующие команды.

3. Вторая программа загружает динамические библиотеки во время выполнения. Пользователь может переключаться между реализациями библиотек и вызывать функции для вычисления числа π и числа x.

Пример работы:

```
./main_dynamic

Dynamic linking example. Enter '0' to switch between lib1 and lib2.

Initially loading lib1.

Commands:

0          -> switch library (lib1 <-> lib2)

1 K        -> calculates Pi(K)

2 x        -> translates x

1 1000

Pi(1000) = 3.140593

2 5

translation(5) = 101

0

Switched to ./liblib2.so

1 1000

Pi(1000) = 3.140807

2 6

translation(6) = 20
```

Выводы: В ходе выполнения лабораторной работы были созданы две динамические библиотеки, реализующие два контракта:

1. Расчёт числа π (Пи) двумя способами: по ряду Лейбница и по формуле Валлиса.
2. Перевод числа x из десятичной системы счисления в двоичную и троичную системы.

Были реализованы две программы для работы с созданными библиотеками:

1. Первая программа использует функции библиотек на этапе компиляции, демонстрируя преимущества статической линковки.
2. Вторая программа загружает библиотеки динамически во время выполнения, обеспечивая гибкость переключения между реализациями.

Выполненные задачи позволили сделать следующие выводы:

Статическая линковка обеспечивает более высокую производительность программы, так как подключение библиотек осуществляется на этапе компиляции.

Динамическая загрузка библиотек позволяет изменять функциональность программы без её повторной компиляции, что особенно удобно для приложений, требующих гибкости и расширяемости.

Реализованная возможность переключения между реализациями функций в динамически загружаемых библиотеках позволяет сравнивать производительность или функциональность различных подходов.

Полученные результаты подтвердили преимущества использования динамических библиотек и соответствуют теоретическим ожиданиям.

.