

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»

Группа: М8О-209БВ-24

Студент: Забродин Р.У.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 23.12.25

Москва, 2025

Постановка задачи

Вариант29

Я разработал две программы, использующие математические функции (вычисление числа π и перевод систем счисления) через динамические библиотеки: первая программа линкуется с библиотекой статически на этапе компиляции, используя ряд Лейбница для вычисления π и двоичную систему для перевода чисел; вторая — загружает реализации динамически во время выполнения, используя `dlopen/dlsym` для переключения между алгоритмами (ряд Лейбница/формула Валлиса для π и двоичная/троичная система для перевода), что демонстрирует различные подходы к интеграции внешнего кода и позволяет сравнивать производительность и точность разных реализаций без перекомпиляции основной программы.

Общий метод и алгоритм решения

Использованные системные вызовы:

void `dlopen(const char filename, int flags)` - загрузка динамической библиотеки в память процесса

void *`dlsym(void handle, const char symbol)` - получение адреса функции по её имени из загруженной библиотеки

int `dlclose(void handle)` - выгрузка библиотеки из памяти процесса

char `dlerror(void)` - получение текстового описания последней ошибки при работе с динамическими библиотеками

Код программы

program1.c

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<../src/
math_utils.h"
```

```

int main() {
    printf("Программа с линковкой на этапе компиляции\n");
    printf("Используется реализация 1 (Лейбница, двоичная система)\n");
    printf("Доступные команды:\n");
    printf(" 0 - информация о реализации\n");
    printf(" 1 K - вычисление Pi с длиной ряда K\n");
    printf(" 2 N - перевод числа N в двоичную систему\n");
    printf(" exit - выход из программы\n\n");

    char command[256];

    while (1) {
        printf("> ");

        if (fgets(command, sizeof(command), stdin) == NULL) {
            break;
        }

        // Удаляем символ новой строки
        command[strcspn(command, "\n")] = '\0';

        if (strcmp(command, "exit") == 0) {
            break;
        }

        if (strcmp(command, "0") == 0) {
            printf("Используется реализация 1:\n");
            printf(" - Pi вычисляется по ряду Лейбница\n");
            printf(" - Перевод в двоичную систему счисления\n");
            continue;
        }

        if (command[0] == '1' && command[1] == ' ') {
            int K;
            if (sscanf(command + 2, "%d", &K) == 1) {
                if (K > 0) {
                    float result = Pi(K);
                    printf("Pi(%d) = %.10f\n", K, result);
                } else {
                    printf("Ошибка: K должно быть положительным числом\n");
                }
            } else {
                printf("Ошибка: неверный формат команды\n");
            }
            continue;
        }

        if (command[0] == '2' && command[1] == ' ') {
            long N;
            if (sscanf(command + 2, "%ld", &N) == 1) {
                char* result = translation(N);
                printf("%ld (10) = %s (2)\n", N, result);
                free(result);
            } else {
                printf("Ошибка: неверный формат команды\n");
            }
            continue;
        }

        printf("Неизвестная команда. Введите '0', '1 K', '2 N' или 'exit'\n");
    }

    printf("Программа завершена.\n");
    return 0;
}

```

```

    printf("Неизвестная команда. Введите '0', '1 K', '2 N' или
'exit'\n");
}

printf("Программа завершена.\n");
return 0;
}

```

program2.c

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dlfcn.h>
#include "../src/math_utils.h"

typedef float (*PiFunc)(int);
typedef char* (*TransFunc)(long);

int main() {
    void* library_handle = NULL;
    PiFunc pi_func = NULL;
    TransFunc trans_func = NULL;

    int current_lib = 1; // 1 - первая реализация, 2 - вторая

    printf("Программа с загрузкой библиотек во время выполнения\n");
    printf("Доступные команды:\n");
    printf(" 0 - переключить реализацию (сейчас: %d)\n", current_lib);
    printf(" 1 K - вычисление Pi с длиной ряда K\n");
    printf(" 2 N - перевод числа N в другую систему\n");
    printf(" exit - выход из программы\n\n");

    // Загружаем первую библиотеку
    library_handle = dlopen("./libmath1.so", RTLD_LAZY);
    if (!library_handle) {
        fprintf(stderr, "Ошибка загрузки библиотеки: %s\n", dlerror());
        return 1;
    }

    // Получаем указатели на функции
    pi_func = (PiFunc)dlsym(library_handle, "Pi");
    trans_func = (TransFunc)dlsym(library_handle, "translation");

    if (!pi_func || !trans_func) {
        fprintf(stderr, "Ошибка получения функций: %s\n", dlerror());
        dlclose(library_handle);
        return 1;
    }

    char command[256];

```

```

char command[256];

while (1) {
    printf("> ");

    if (fgets(command, sizeof(command), stdin) == NULL) {
        break;
    }

    // Удаляем символ новой строки
    command[strcspn(command, "\n")] = '\0';

    if (strcmp(command, "exit") == 0) {
        break;
    }

    if (strcmp(command, "0") == 0) {
        // Переключаем библиотеку
        dlclose(library_handle);

        if (current_lib == 1) {
            current_lib = 2;
            library_handle = dlopen("./libmath2.so", RTLD_LAZY);
            if (!library_handle) {
                fprintf(stderr, "Ошибка загрузки библиотеки 2: %s\n", dlerror());
                return 1;
            }
            printf("Переключено на реализацию 2 (Валлис, троичная система)\n");
        } else {
            current_lib = 1;
            library_handle = dlopen("./libmath1.so", RTLD_LAZY);
            if (!library_handle) {
                fprintf(stderr, "Ошибка загрузки библиотеки 1: %s\n", dlerror());
                return 1;
            }
            printf("Переключено на реализацию 1 (Лейбница, двоичная система)\n");
        }
    }

    // Получаем указатели на функции из новой библиотеки
    pi_func = (PiFunc)dlsym(library_handle, "Pi");
    trans_func = (TransFunc)dlsym(library_handle, "translation");

    if (!pi_func || !trans_func) {
        fprintf(stderr, "Ошибка получения функций: %s\n", dlerror());
        dlclose(library_handle);
        return 1;
    }

    continue;
}

if (command[0] == '1' && command[1] == ' ') {
    int K;
    if (sscanf(command + 2, "%d", &K) == 1) {
        if (K > 0) {
            float result = pi_func(K);
            printf("Pi(%d) = %.10f (реализация %d)\n", K, result, current_lib);
        } else {
            printf("Ошибка: K должно быть положительным числом\n");
        }
    } else {
        printf("Ошибка: неверный формат команды\n");
    }
    continue;
}

```

```

if (command[0] == '2' && command[1] == ' ') {
    long N;
    if (sscanf(command + 2, "%ld", &N) == 1) {
        char* result = trans_func(N);
        if (current_lib == 1) {
            printf("%ld (10) = %s (2)\n", N, result);
        } else {
            printf("%ld (10) = %s (3)\n", N, result);
        }
        free(result);
    } else {
        printf("Ошибка: неверный формат команды\n");
    }
    continue;
}

printf("Неизвестная команда. Введите '0', '1 K', '2 N' или 'exit\n");
}

// Закрываем библиотеку
if (library_handle) {
    dlclose(library_handle);
}

printf("Программа завершена.\n");
return 0;
}

```

pi_libniz.c

```

#include "../src/math_utils.h"
#include <stdlib.h>
float Pi(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) {
    if (x == 0) {
        char* result = (char*)malloc(2 * sizeof(char));
        result[0] = '0';
        result[1] = '\0';
        return result;
    }
    long temp = x;
    int length = 0;
    while (temp > 0) {
        temp /= 2;
        length++;
    }
    char* result = (char*)malloc((length + 1) * sizeof(char));
    int index = length - 1;
    temp = x;
    while (temp > 0) {
        result[index] = (temp % 2) + '0';
        temp /= 2;
        index--;
    }
    result[length] = '\0';
    return result;
}

```

pi_wallis.c

```
#include "../../src/math_utils.h"
#include <stdlib.h>

float Pi(int K) {
    float pi = 1.0;

    for (int i = 1; i <= K; i++) {
        float factor = (4.0 * i * i) / (4.0 * i * i - 1);
        pi *= factor;
    }

    return pi * 2;
}

char* translation(long x) {
    if (x == 0) {
        char* result = (char*)malloc(2 * sizeof(char));
        result[0] = '0';
        result[1] = '\0';
        return result;
    }

    // Определяем размер необходимой памяти
    long temp = x;
    int length = 0;

    while (temp > 0) {
        temp /= 3;
        length++;
    }

    // Выделяем память под результат
    char* result = (char*)malloc((length + 1) * sizeof(char));

    // Заполняем строку с конца
    int index = length - 1;
    temp = x;

    while (temp > 0) {
        int digit = temp % 3;
        result[index] = digit + '0';
        temp /= 3;
        index--;
    }

    result[length] = '\0';
    return result;
}
```

Протокол работы программы

```
echo -e "1 10\n1 100\n1 1000\n1 10000\nexit" | ./program1
```

Программа с линковкой на этапе компиляции

Используется реализация 1 (Лейбниц, двоичная система)

Доступные команды:

0 - информация о реализации

1 K - вычисление Pi с длиной ряда K

2 N - перевод числа N в двоичную систему

exit - выход из программы

```
> Pi(10) = 3.0418398380
```

```
> Pi(100) = 3.1315927505
```

```
> Pi(1000) = 3.1405928135
```

```
> Pi(10000) = 3.1414983273
```

```
> Программа завершена.
```

```
echo -e "0\n1 10\n1 100\n1 1000\n1 10000\nexit" | LD_LIBRARY_PATH=. ./program2
```

Программа с загрузкой библиотек во время выполнения

Доступные команды:

0 - переключить реализацию (сейчас: 1)

1 K - вычисление Pi с длиной ряда K

2 N - перевод числа N в другую систему

exit - выход из программы

```
> Переключено на реализацию 2 (Валлис, троичная система)
```

```
> Pi(10) = 3.0677049160 (реализация 2)
```

```
> Pi(100) = 3.1337878704 (реализация 2)
```

```
> Pi(1000) = 3.1408064365 (реализация 2)
```

```
> Pi(10000) = 3.1413495541 (реализация 2)
```

```
> Программа завершена.
```

```
echo -e "2 0\n2 1\n2 5\n2 10\n2 255\n2 1024\nexit" | ./program1
```

Программа с линковкой на этапе компиляции

Используется реализация 1 (Лейбниц, двоичная система)

Доступные команды:

0 - информация о реализации

1 K - вычисление Pi с длиной ряда K

2 N - перевод числа N в двоичную систему

exit - выход из программы

```
> 0 (10) = 0 (2)
```

```
> 1 (10) = 1 (2)
```

```
> 5 (10) = 101 (2)
```

```
> 10 (10) = 1010 (2)
```

```
> 255 (10) = 11111111 (2)
```

```
> 1024 (10) = 10000000000 (2)
```

```
> Программа завершена.
```

```
echo -e "0\n2 0\n2 1\n2 5\n2 10\n2 27\n2 81\nexit" | LD_LIBRARY_PATH=. ./program2
```

Программа с загрузкой библиотек во время выполнения

Доступные команды:

0 - переключить реализацию (сейчас: 1)

1 K - вычисление Pi с длиной ряда K

2 N - перевод числа N в другую систему

exit - выход из программы

```
> Переключено на реализацию 2 (Валлис, троичная система)
```

```
> 0 (10) = 0 (3)
```

```
> 1 (10) = 1 (3)
```

```
> 5 (10) = 12 (3)
```

```
> 10 (10) = 101 (3)
```

```
> 27 (10) = 1000 (3)
```

```
> 81 (10) = 10000 (3)
```

```
> Программа завершена.
```


Вывод

Данная лабораторная работа демонстрирует механизм динамической загрузки библиотек (dynamic loading) во время выполнения программы. Основная программа интерактивно загружает различные реализации математических функций из внешних разделяемых объектов (`libimpl1.so` и `libimpl2.so`), используя функции `dlopen`, `dlSym` и `dlClose`. Реализация позволяет переключаться между альтернативными алгоритмами вычисления числа Пи и перевода числа x в различные системы счисления

. Работа подтверждает эффективность подхода pluginной архитектуры для создания модульных и расширяемых приложений, где функциональность может быть изменена простой заменой динамических библиотек.