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

Разработать программу на языке C, вычисляющую интеграл функции sin(x) на заданном отрезке [A, B] с указанным шагом e и сортирующую целочисленный массив. Реализовать две версии алгоритмов для каждой операции: для интегрирования — метод прямоугольников и метод трапеций; для сортировки — пузырьковую сортировку и быструю сортировку Хоара. Оформить каждую пару алгоритмов в виде отдельной динамической библиотеки (.so). Создать две демонстрационные программы: первая должна использовать одну из реализаций путём статической линковки на этапе компиляции, вторая — динамически загружать библиотеки во время выполнения с возможностью их переключения по команде пользователя.

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

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.

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 (Лейбниц, 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 (Лейбниц, 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

```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.
```

```
strace -f ./program2
execve("./program2", ["./program2"], 0x7ffc3db04c98 /* 36 vars */) = 0
brk(NULL)               = 0x60f2afe0b000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7a8d06a2e000
access("/etc/ld.so.preload", R_OK)   = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=19827, ...}) = 0
mmap(NULL, 19827, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7a8d06a29000
close(3)                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784


mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7a8d06800000
mmap(0x7a8d06828000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000) = 0x7a8d06828000
mmap(0x7a8d069b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) =
0x7a8d069b0000
mmap(0x7a8d069ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000)
= 0x7a8d069ff000
mmap(0x7a8d06a05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7a8d06a05000
close(3)                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7a8d06a26000
arch_prctl(ARCH_SET_FS, 0x7a8d06a26740) = 0
set_tid_address(0x7a8d06a26a10)     = 48579
set_robust_list(0x7a8d06a26a20, 24)   = 0
rseq(0x7a8d06a27060, 0x20, 0, 0x53053053) = 0
mprotect(0x7a8d069ff000, 16384, PROT_READ) = 0
mprotect(0x60f29678b000, 4096, PROT_READ) = 0
mprotect(0x7a8d06a66000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7a8d06a29000, 19827)      = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...}) = 0
getrandom("\xdb\x25\x55\x92\x6b\x09\xe9\xa3", 8, GRND_NONBLOCK) = 8
brk(NULL)               = 0x60f2afe0b000
brk(0x60f2afe2c000)           = 0x60f2afe2c000
write(1, "\320\237\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260 \321\201
\320\267\320\260\320\263\321\200\321\203"..., 97Программа с загрузкой библиотек во время выполнения
) = 97
write(1, "\320\224\320\276\321\201\321\202\321\203\320\277\320\275\321\213\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321"..., 35Доступные команды:
) = 35
write(1, " 0 - \320\277\320\265\321\200\320\265\320\272\320\273\321\216\321\207\320\270\321\202\321\214 \321\200\320"...,
68 0 - переключить реализацию (сейчас: 1)
) = 68
write(1, " 1 K - \320\262\321\213\321\207\320\270\321\201\320\273\320\265\320\275\320\270\320\265 Pi "..., 59 1 K -
вычисление Pi с длиной ряда K
) = 59
write(1, " 2 N - \320\277\320\265\321\200\320\265\320\262\320\276\320\264 \321\207\320\270\321\201\320\273\320"..., 67 2
N - перевод числа N в другую систему
) = 67
write(1, " exit - \320\262\321\213\321\205\320\276\320\264 \320\270\320\267 \320\277\321\200\320\276\320"..., 44 exit -
выход из программы
) = 44
write(1, "\n", 1
)           = 1
```

```
openat(AT_FDCWD, "./libmath1.so", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=15584, ...}) = 0
getcwd("/home/lolol/projects/6laba/4lab/src", 128) = 36
mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x780648aab000
mmap(0x780648aac000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000)
= 0x780648aac000
mmap(0x780648aad000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x780648aad000
mmap(0x780648aae000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x780648aae000
close(3)                = 0
mprotect(0x780648aae000, 4096, PROT_READ) = 0
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...}) = 0
write(1, "> ", 2> )             = 2
read(0,
```

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 плагинной

архитектуры для создания модульных и расширяемых приложений, где функциональность может

быть изменена простой заменой динамических библиотек.