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-211БВ-24

Студент: ПРОХОРОВ В.И.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 17.02.26

Москва, 2026

Постановка задачи

Вариант 25.

Функция 1 - 4 вариант

Функция 2 - 8 вариант

4 Вариант:

Подсчёт наибольшего общего делителя для двух натуральных

чисел:

Сигнатура функции: int gcd(int a, int b);

- Реализация №1: Алгоритм Евклида

- Реализация №2: Наивный алгоритм: пытаться разделить числа на все числа, что меньше a и b.

8 Вариант:

Перевод числа x из десятичной системы счисления в другую:

Сигнатура функции: char *convert(int x);

- Реализация №1: Перевод в двоичную

- Реализация №2: Перевод в троичную

Общий метод и алгоритм решения

Использованные вызовы:

- **void *dlopen(const char *filename, int flags);** - Открывает (загружает в память процесса) динамическую библиотеку (.so файл) и возвращает дескриптор (handle), который потом используется в других функциях.
- **void *dlsym(void *handle, const char *symbol);** - Ищет в уже загруженной библиотеке (по дескриптору handle) символ с именем symbol и возвращает указатель на него.
- **int dlclose(void *handle);** - Выгружает (закрывает) ранее открытую библиотеку из памяти процесса.

Идея решения:

Имеются 4 библиотеки, каждая из которых содержит свою функцию в своей имплементации либо gcd, либо convert, в первой тестовой программе test_static.c библиотеки используются во время компиляции (на этапе линковки/linking) и использует функции из двух библиотек, в моём случае (gcd1, convert1). Во время исполнения второй программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками (dlfcn). Изначально библиотека загружается **dlopen**, потом ищутся функции (gcd, convert) с помощью **dlsym**, если находятся то используем их, если нет то есть заглушки которые не роняют программу. И после закрываем библиотеки с помощью **dlclose**. Смена библиотек происходит при вводе команды 0.

Код программы

test_static.c

```c
#include "library.h"
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    char line[256];

    while (fgets(line, sizeof(line), stdin)) {
        int cmd;
        if (sscanf(line, "%d", &cmd) != 1)
            continue;

        if (cmd == 1) {
            int a, b;
            if (sscanf(line, "%*d %d %d", &a, &b) == 2) {
                printf("%d\n", gcd(a, b));
            }
        } else if (cmd == 2) {
            int x;
            if (sscanf(line, "%*d %d", &x) == 1) {
                char *res = convert(x);
                if (res) {
                    printf("%s\n", res);
                    free(res);
                } else {
                    printf("error\n");
                }
            }
        }
    }
```

```
    return 0;
}
```

Test_dynamic.c

```
#include "library.h"
#include <dlfcn.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// указатели на функции
static gcd_func_t *gcd_ptr = NULL;
static convert_func_t *convert_ptr = NULL;

// дискрипторы библиотек
static void *handle_gcd = NULL;
static void *handle_conv = NULL;

static int current = 1; // 1 или 2

// заглушки
static int gcd_stub(int a, int b) {
    (void)a;
    (void)b;
    return 0;
}

static char *convert_stub(int x) {
    (void)x;
    return strdup("error: library not loaded");
}

void load_libraries(void) {
    char path_g[64];
    char path_c[64];

    // формируем имена библиотек
    snprintf(path_g, sizeof(path_g), "./libgcd%d.so", current);
    snprintf(path_c, sizeof(path_c), "./libconvert%d.so", current);

    // закрываем библиотеки если они были открыты ранее
    if (handle_gcd)
        dlclose(handle_gcd);
    if (handle_conv)
        dlclose(handle_conv);
```

```c
    // загружаем библиотеки
    handle_gcd = dlopen(path_g, RTLD_NOW | RTLD_LOCAL);
    handle_conv = dlopen(path_c, RTLD_NOW | RTLD_LOCAL);

    // пытаемся найти функции gcd/convert
    gcd_ptr = handle_gcd ? dlsym(handle_gcd, "gcd") : NULL;
    convert_ptr = handle_conv ? dlsym(handle_conv, "convert") : NULL;

    // если не нашли используем заглушки
    if (!gcd_ptr)
        gcd_ptr = gcd_stub;
    if (!convert_ptr)
        convert_ptr = convert_stub;
}

int main(void) {
    load_libraries();

    char line[256];

    while (fgets(line, sizeof(line), stdin)) {
        int cmd;
        if (sscanf(line, "%d", &cmd) != 1)
            continue;

        if (cmd == 0) {
            current = 3 - current;
            load_libraries();
            printf("Switched to implementation %d\n", current);
            continue;
        }

        if (cmd == 1) {
            int a, b;
            if (sscanf(line, "%*d %d %d", &a, &b) == 2) {
                printf("%d\n", gcd_ptr(a, b));
            }
        } else if (cmd == 2) {
            int x;
            if (sscanf(line, "%*d %d", &x) == 1) {
                char *res = convert_ptr(x);
                if (res) {
                    printf("%s\n", res);
                    free(res);
                } else {
                    printf("error\n");
                }
```

```
            }
        }
    }

    if (handle_gcd)
        dlclose(handle_gcd);
    if (handle_conv)
        dlclose(handle_conv);

    return 0;
}
```

Протокол работы программы

```
→  lab_4 git:(main) ✗ ./test_static
1 10 2
2
2 10
1010
1 3 6
3
2 21
10101
```

```
→  lab_4 git:(main) ✗ ./test_dynamic
1 10 2
2
2 10
1010
0
Switched to implementation 2
1 10 2
2
2 10
101
0
Switched to implementation 1
1 21 7
7
2 21
10101
```

Вывод

При выполнении данной лабораторной работы я научился создавать динамические библиотеи, а так же писать программы, которые используют функции динамических библиотек. При подключении библиотек на этапе компилирования (линковки), программа быстрее запускается (не надо загружать библиотеки), происходит меньше ошибок в runtime все зависимости проверяются на этапе линковки, а так-же работает компиляторная оптимизация. Из минусов меньшая гибкость (невозможность сменить

имплементацию функции без перекомпилирования программы), зависимость от библиотек, и (скорее всего) функция не обновляется даже если вышла новая версия библиотеки. При загрузке библиотек во время runtime библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками, у этого метода есть свои плюсы: можно менять реализацию функции без перекомпиляции (нужно для команды ‘0’), большая модульность, а так-же некоторая экономия памяти, так-как библиотека загружается по необходимости. Из минусов же медленный запуск из за overhead (подключение библиотек и поиск функций), риск ошибок в runtime (если библиотека не найдена к примеру), и более сложная отладка из-за того что ошибки проявляются только при запуске.

Если библиотеку подключать во время компиляции, то она работает быстрее, но ниже гибкость. А если во время исполнения программы, то наоборот, ниже стабильность но выше гибкость.