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»

Выполнил: А. А. Устинов
Группа: М8О-207БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

Цель работы: Приобретение практических навыков в создании динамических библиотек и в создании программ, которые используют функции динамических библиотек

Задание: Требуется создать динамические библиотеки, которые реализуют заданный вариантом функционал. Далее использовать данные библиотеки 2-мя способами: 1. Во время компиляции (на этапе «линковки»/linking) 2. Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками В конечном итоге, в лабораторной работе необходимо получить следующие части: Динамические библиотеки, реализующие контракты, которые заданы вариантом; Тестовая программа (программа №1), которая используют одну из библиотек, используя информацию полученные на этапе компиляции; Тестовая программа (программа №2), которая загружает библиотеки, используя только их относительные пути и контракты. Провести анализ двух типов использования библиотек.

Вариант: 28

Метод решения

Задача состояла в создании динамических библиотек, реализующих заданный функционал (Вариант 28), и создании двух тестовых программ, демонстрирующих разные методы использования этих библиотек.

Контракты

Для обеспечения совместимости между всеми модулями (двумя библиотеками и двумя программами) был разработан единый контракт, определенный в файле `contract.h`.

1. **Функция 1 (Рассчет π):** `float Pi(int K)`
    - Реализация 1 (`liblib1.so`): Ряд Лейбница.
    - Реализация 2 (`liblib2.so`): Формула Валлиса.
2. **Функция 2 (Площадь):** `float Square(float A, float B)`
    - Реализация 1 (`liblib1.so`): Площадь прямоугольника ($A \cdot B$).
    - Реализация 2 (`liblib2.so`): Площадь прямоугольного треугольника ($\frac{1}{2}A \cdot B$).

Методы Использования Библиотек

1. **Программа №1 (Ранняя Линковка):** Исполняемый файл `prog1` линкуется с `liblib1.so` на этапе компиляции с использованием флага `-l`. Зависимость фиксируется в заголовке программы.
2. **Программа №2 (Поздняя Линковка):** Исполняемый файл `prog2` использует системный интерфейс (функции `dlopen`, `dlsym`, `dlclose` из `dlfcn.h` с линковкой `-ldl`) для загрузки и выгрузки `liblib1.so` или `liblib2.so` непосредственно во время работы программы.

Архитектура программы и Структура файлов

```
lab4/
├── inc/
│   └── contract.h
├── src/
│   ├── lib1.c
│   ├── lib2.c
│   ├── main_static.c
│   └── main_dynamic.c
└── Makefile
```

Описание Программных Модулей

Файл `inc/contract.h`

Определяет сигнатуры функций `Pi(int K)` и `Square(float A, float B)`, создавая общий интерфейс, который должны реализовать обе динамические библиотеки.

Файл `src/lib1.c` (Реализация 1)

Реализует функции контракта, используя Ряд Лейбница для расчета π и формулу площади прямоугольника. Компилируется с флагами `-fPIC` (Position Independent Code) и `-shared` для создания динамической библиотеки `liblib1.so`.

Файл `src/lib2.c` (Реализация 2)

Реализует функции контракта, используя Формулу Валлиса для расчета π и формулу площади прямоугольного треугольника. Компилируется с флагами `-fPIC` и `-shared` для создания динамической библиотеки `liblib2.so`.

Файл `src/main_static.c` (Программа №1)

Тестовая программа, демонстрирующая использование библиотеки через раннюю линковку.

Логика:

- Осуществляет прямой вызов функций `Pi()` и `Square()`.
- Программа `prog1` при сборке линкуется с `liblib1.so` и всегда использует только эту реализацию.
- Ввод команд организован по протоколу: `1 arg1...` для `Pi`, `2 arg1 arg2...` для `Square`.

Файл `src/main_dynamic.c` (Программа №2)

Назначение: Тестовая программа, демонстрирующая использование библиотеки через **позднюю линковку** и переключение реализаций.

Логика:

- Использует системный API: `dlopen` (загрузка библиотеки), `dlsym` (получение адреса функции) и `dlclose` (выгрузка).
- Вызов функций `Pi` и `Square` осуществляется через указатели на функции, полученные из `dlsym`.
- Команда `0` реализована для выгрузки текущей библиотеки и загрузки альтернативной, обеспечивая динамическую смену функционала во время исполнения.

Результаты

Разработанное решение представляет собой систему из двух динамических библиотек (`liblib1.so`, `liblib2.so`) и двух исполняемых программ (`prog1`, `prog2`), демонстрирующих два метода линковки и загрузки кода.

Проверка Программы №1 (Ранняя Линковка)

Было подтверждено, что программа `prog1` была успешно слинкована с `liblib1.so` на этапе компиляции. Вызовы функций `Pi(K)` и `Square(A, B)` всегда используют Реализацию 1(Ряд Лейбница и Площадь прямоугольника). Присутствие `liblib1.so` является критически важным для запуска программы.

Проверка Программы №2 (Поздняя Линковка)

Было подтверждено, что программа `prog2` успешно использует системный интерфейс `dlfcn.h` для динамической загрузки библиотек.

Программа корректно загружает `liblib1.so` при старте. Команда `0` успешно выполняет выгрузку текущей библиотеки и загрузку `liblib2.so` (или наоборот), демонстрируя переключение контракта во время исполнения. После переключения функции `Pi()` и `Square()` демонстрируют новое поведение Реализация 2: Формула Валлиса и Площадь прямоугольного треугольника).

Пример взаимодействия

Ниже приведен пример сессии взаимодействия с Программой №2, демонстрирующий смену реализации во время работы.

Сессия:

Листинг 1: Сессия работы с prog2

```
$ ./prog2
Dynamic Loading
[...]
```

```
Succes to load Program 1.

>>> 2 10 5
Square(10.00, 5.00) [Rectangle] = 50.00

>>> 0
Succes to load Program 2.

>>> 2 10 5
Square(10.00, 5.00) [Rect triangle] = 25.00

>>> 1 1000
Pi(1000) [Walis] = 3.14159049

>>> exit
```

Ключевые особенности

1. **Инкапсуляция Реализаций:** Разработаны две полностью независимые динамические библиотеки (`liblib1.so`, `liblib2.so`), каждая из которых реализует один и тот же контракт, что позволяет легко менять функционал без изменения основного кода.

2. **Демонстрация Ранней Линковки:** Программа `prog1` подтверждает механизм **статической (ранней)** линковки, где зависимость разрешается загрузчиком ОС при запуске.

3. **Динамическая Загрузка:** Программа `prog2` использует функции `dlopen`, `dlsym` и `dlclose`, демонстрируя **позднюю** линковку и возможность управления модулями из кода приложения.

4. **Переключение Контракта:** Реализована команда `0`, которая обеспечивает смену всей логики вычислений (Pi и Square) во время исполнения, демонстрируя возможности архитектуры плагинов.

5. **Независимость Кода:** Исходный код `main_dynamic.c` не содержит ссылок на конкретные библиотеки, что делает его гибким и независимым от выбранной реализации.

Выводы

В ходе выполнения лабораторной работы были успешно освоены методы создания и использования динамических библиотек на языке C в среде Linux. Были разработаны две независимые динамические библиотеки (`liblib1.so` и `liblib2.so`), реализующие единый контракт (`contract.h`), что позволило обеспечить возможность замены функционала. На практике был изучен механизм ранней (статической) линковки (`prog1`), при которой зависимость от `liblib1.so` фиксируется на этапе сборки и управляется системным загрузчиком, демонстрируя сильную зависимость исполняемого файла от присутствия библиотеки. Параллельно был освоен механизм поздней (динамической) загрузки с использованием интерфейса `dlopen/dlsym/dlclose` в Программе №2. Эта программа

продемонстрировала гибкость и слабую зависимость, самостоятельно управляя жизненным циклом библиотек. Ключевой результат был достигнут в Программе №2, где реализованная команда 0 позволила на лету переключать всю логику вычислений (Pi и Square) между Реализацией 1 и Реализацией 2. Полученные результаты подтверждают, что поздняя линковка является мощным инструментом для создания модульных и расширяемых приложений (систем плагинов), где функционал может быть изменен или обновлен без необходимости перекомпиляции основного исполняемого файла. Структура проекта, разделенная на контракты, реализации и тестовые программы, обеспечила ясное разделение ответственности между модулями.

Исходная программа

Файл lib1.c

```c
#include <math.h>

#include "../inc/contract.h"

float Pi(int K) {
    float pi_approx = 0.0f;
    for (int n = 0; n < K; n++) {
        if (n % 2 == 0) {
            pi_approx += 1.0f / (2.0f * n + 1.0f);
        } else {
            pi_approx -= 1.0f / (2.0f * n + 1.0f);
        }
    }
    return 4.0f * pi_approx;
}

float Square(float A, float B) {
    return A * B;
}
```

Листинг 2: Ряд Лейбница для и площадь прямоугольника.

Файл lib2.c

```c
#include <stdio.h>

#include "../inc/contract.h"


float Pi(int K) {
    if (K <= 0) return 0.0f;
    float pi_half_approx = 1.0f;
    for (int n = 1; n <= K; n++) {
        pi_half_approx *= (2.0f * n) / (2.0f * n - 1.0f);
        pi_half_approx *= (2.0f * n) / (2.0f * n + 1.0f);
    }
    return 2.0f * pi_half_approx;
}

```

```c
float Square(float A, float B) {
    return 0.5f * A * B;
}
```

Листинг 3: Формула Валлиса для и площадь прямоугольного треугольника.

Файл main_dynamic.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dlfcn.h>


typedef float (*Pi_func)(int);
typedef float (*Square_func)(float, float);

void *lib_handle = NULL;
Pi_func pi_func_ptr = NULL;
Square_func square_func_ptr = NULL;
int current_lib = 0; // 0 - not loaded, 1 - lib1, 2 - lib2

const char *LIB_PATH_1 = "./liblib1.so";
const char *LIB_PATH_2 = "./liblib2.so";



int load_library(int lib_id) {
    const char *lib_path = (lib_id == 1) ? LIB_PATH_1 : LIB_PATH_2;

    if (lib_handle != NULL) {
        dlclose(lib_handle);
        lib_handle = NULL;
        pi_func_ptr = NULL;
        square_func_ptr = NULL;
    }

    lib_handle = dlopen(lib_path, RTLD_LAZY);
    if (!lib_handle) {
        fprintf(stderr, "Error load library %s: %s\n", lib_path, dlerror());
        current_lib = 0;
        return 0;
    }

    pi_func_ptr = (Pi_func)dlsym(lib_handle, "Pi");
    if (dlerror() != NULL) {
        fprintf(stderr, "Error to find Pi in %s: %s\n", lib_path, dlerror());
        dlclose(lib_handle);
        lib_handle = NULL;
        current_lib = 0;
        return 0;
    }

    square_func_ptr = (Square_func)dlsym(lib_handle, "Square");
    if (dlerror() != NULL) {
        fprintf(stderr, "Error to find Square in %s: %s\n", lib_path, dlerror());
```

```
49          dlclose(lib_handle);
50          lib_handle = NULL;
51          current_lib = 0;
52          return 0;
53      }
54
55      current_lib = lib_id;
56      printf("Success! %d.\n", current_lib);
57      return 1;
58  }
59
60  void process_command(int func_id, char* args[]) {
61      if (lib_handle == NULL) {
62          printf("Library not loaded\n");
63          return;
64      }
65
66      if (func_id == 1) {
67          if (args[0] == NULL) {
68              printf("Func 1 need 1 argument(K).\n");
69              return;
70          }
71          int K = atoi(args[0]);
72          if (K <= 0) {
73              printf("K have to be positive\n");
74              return;
75          }
76
77
78          float result = pi_func_ptr(K);
79          const char *impl_name = (current_lib == 1) ? "L" : "V";
80          printf("Pi(%d) [%s] = %.8f\n", K, impl_name, result);
81      }
82
83      else if (func_id == 2) {
84          if (args[0] == NULL || args[1] == NULL) {
85              printf("Func 2 need 2 arguments\n");
86              return;
87          }
88          float A = atof(args[0]);
89          float B = atof(args[1]);
90
91          float result = square_func_ptr(A, B);
92          const char *impl_name = (current_lib == 1) ? "Rectangle" : "Rect triangle";
93          printf("Square(%.2f, %.2f) [%s] = %.2f\n", A, B, impl_name, result);
94      } else {
95          printf("Unknown.\n");
96      }
97  }
98
99  int main() {
100     char line[256];
101     printf("Commands:\n");
102     printf("0 : Realisation swap (1 <-> 2).\n");
103     printf("1 arg1...: Pi(K).\n");
104     printf("2 arg1 arg2...: Square(A, B).\n");
105
106     if (!load_library(1)) {
```

```
        fprintf(stderr, "Error to load libraries\n");
        return 1;
    }

    while (1) {
        if (fgets(line, sizeof(line), stdin) == NULL) break;

        char *token;
        char *tokens[3] = {NULL, NULL, NULL};
        int token_count = 0;

        token = strtok(line, " \n");
        while (token != NULL && token_count < 3) {
            tokens[token_count++] = token;
            token = strtok(NULL, " \n");
        }

        if (token_count == 0 || strcmp(tokens[0], "exit") == 0) break;

        if (token_count >= 1) {
            int command = atoi(tokens[0]);

            if (command == 0) {
                int next_lib = (current_lib == 1) ? 2 : 1;
                load_library(next_lib);
            } else {
                process_command(command, &tokens[1]);
            }
        }
    }

    if (lib_handle != NULL) {
        dlclose(lib_handle);
    }
    return 0;
}
```

Листинг 4: Динамическая загрузка

Файл main_static.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "../inc/contract.h"


void process_command(int func_id, char* args[]) {
    if (func_id == 1) {
        if (args[0] == NULL) {
            printf("Error(K).\n");
            return;
        }
        int K = atoi(args[0]);
        if (K <= 0) {
```

```
            printf("Error\n");
            return;
        }
        float result = Pi(K);
        printf("Result Pi(%d) = %.8f\n", K, result);
    }

    else if (func_id == 2) {
        if (args[0] == NULL || args[1] == NULL) {
            printf("Error (A, B).\n");
            return;
        }
        float A = atof(args[0]);
        float B = atof(args[1]);

        float result = Square(A, B);
        printf("Square(%.2f, %.2f) = %.2f\n", A, B, result);
    } else {
        printf("Unknown \n");
    }
}

int main() {
    char line[256];
    printf("Realisation 1 (L / Rectangle)\n");

    while (1) {
        printf(">>> ");
        if (fgets(line, sizeof(line), stdin) == NULL) break;

        char *token;
        char *tokens[3] = {NULL, NULL, NULL};
        int token_count = 0;

        token = strtok(line, " \n");
        while (token != NULL && token_count < 3) {
            tokens[token_count++] = token;
            token = strtok(NULL, " \n");
        }

        if (token_count == 0 || strcmp(tokens[0], "exit") == 0) break;

        if (token_count >= 1) {
            int func_id = atoi(tokens[0]);
            if (func_id != 0) {
                process_command(func_id, &tokens[1]);
            } else {
                printf("Error '0' \n");
            }
        }
    }

    return 0;
}
```

Листинг 5: Статическая линковка

Вывод strace

Для prog1

```
execve("./ipc_program", ["./ipc_program"], 0x7ffcca0c1630 /* 27 vars */) = 0
brk(NULL)                               = 0x5ea37c54b000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x76058ef9
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=30255, ...}) = 0
mmap(NULL, 30255, PROT_READ, MAP_PRIVATE, 3, 0) = 0x76058ef97000
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\0"..., 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, 6
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, 6
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x76058ec00000
mmap(0x76058ec28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRIT
mmap(0x76058edb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0
mmap(0x76058edff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
mmap(0x76058ee05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x76058ef
arch_prctl(ARCH_SET_FS, 0x76058ef94740) = 0
set_tid_address(0x76058ef94a10)         = 1482
set_robust_list(0x76058ef94a20, 24)     = 0
rseq(0x76058ef95060, 0x20, 0, 0x53053053) = 0
mprotect(0x76058edff000, 16384, PROT_READ) = 0
mprotect(0x5ea33ed8e000, 4096, PROT_READ) = 0
mprotect(0x76058efd7000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x76058ef97000, 30255)           = 0
write(2, "\320\230\321\201\320\277\320\276\320\273\321\214\320\267\320\276\320\262\32
) = 53
exit_group(1)                           = ?
```

Для prog2

```
execve("./prog2", ["./prog2"], 0x7ffc688ed910 /* 27 vars */) = 0
brk(NULL)                               = 0x62750638a000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ee77976
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=30255, ...}) = 0
mmap(NULL, 30255, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ee779763000
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\0"..., 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, 6
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, 6
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ee779400000
mmap(0x7ee779428000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRIT
mmap(0x7ee7795b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0
mmap(0x7ee7795ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
mmap(0x7ee779605000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ee7797
arch_prctl(ARCH_SET_FS, 0x7ee779760740) = 0
set_tid_address(0x7ee779760a10)         = 9490
set_robust_list(0x7ee779760a20, 24)     = 0
rseq(0x7ee779761060, 0x20, 0, 0x53053053) = 0
mprotect(0x7ee7795ff000, 16384, PROT_READ) = 0
mprotect(0x6274fb7c5000, 4096, PROT_READ) = 0
mprotect(0x7ee7797a3000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7ee779763000, 30255)           = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
getrandom("\x55\xd2\x7f\xbd\x4f\x65\xd2\xcb", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x62750638a000
brk(0x6275063ab000)                     = 0x6275063ab000
write(1, "\320\232\320\276\320\274\320\260\320\275\320\264\321\213:\n", 16Команды:
) = 16
write(1, "0 : \320\237\320\265\321\200\320\265\320\272\320\273\321\216\321\207\320\26
) = 61
write(1, "1 arg1...: \320\222\321\213\320\267\320\276\320\262 Pi(K).\n", 291 arg1...:
) = 29
write(1, "2 arg1 arg2...: \320\222\321\213\320\267\320\276\320\262 Squar"..., 412 arg
) = 41
openat(AT_FDCWD, "./liblib1.so", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 8
fstat(3, {st_mode=S_IFREG|0755, st_size=15208, ...}) = 0
getcwd("/home/b0rus1a/OSLabs/Lab4", 128) = 26
mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ee779766000
mmap(0x7ee779767000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
mmap(0x7ee779768000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000)
mmap(0x7ee779769000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
close(3)                                = 0
mprotect(0x7ee779769000, 4096, PROT_READ) = 0
write(1, "\320\243\321\201\320\277\320\265\321\210\320\275\320\276 \320\267\320\260\3
) = 58
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, ">>> ", 4>>> )                      = 4
```