Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

**Динамические библиотеки.**

Студент: Мариничев И. А.

Группа: М8О-208Б-19

Преподаватель: Миронов Е. С.

Дата: 17.12.2020

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2020

1. **Постановка задачи. Вариант 32**

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки двумя способами:

* во время компиляции (на этапе линковки)
* во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

В лабораторной работе необходимо получить следующие части:

* Динамические библиотеки, реализующие контракты
* Тестовая программа (программа №1), которая использует одну из библиотек, используя знания, полученные на этапе компиляции.
* Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты.

Провести анализ использования двух типов библиотек.

Пользовательский ввод должен быть организован следующим образом:

* Команда «0»: переключить одну реализацию контрактов на другую
* Команда «1 args»: вызов первой функции контрактов
* Команда «2 args»: вызов второй функции контрактов

Контракты:

* Расчёт значения числа е (основание натурального логарифма) – при помощи второго замечательного предела и при помощи ряда Тейлора.
* Перевод числа x из десятичной сс в другую – реализация 1: в двоичную сс, реализация 2: в троичную сс.

1. **Общие сведения о программе**

Программа написана на языке Си в UNIX-подобной операционной системе (Ubuntu).

Контракты описаны в заголовочном файле lib/implementation.h. Реализации контрактов описаны в файлах lib/implementation-1.c и lib/implementation-2.c.

Сборка программы производится с использованием Makefile. Для подключения библиотеки на этапе компиляции выполняются следующие действия:

1. Получение объектного файла основной программы.
2. Компиляция файла библиотеки с ключом -shared. Получаем .so файл.
3. Линковка библиотеки и объектного файла основной программы.
4. Указание пути к библиотеке в переменной LD\_LIBRARY\_PATH.

export LD\_LIBRARY\_PATH=.

Для динамической загрузки библиотек используются средства библиотеки dlfcn.h.

В программе используются следующие системные вызовы:

1. **dlopen** – загружает динамическую библиотеку с указанным именем. В случае неуспеха возвращает NULL.
2. **dlclose –** уменьшает на единицу счетчик ссылок на указатель динамической библиотеки *handle*. Если нет других загруженных библиотек, использующих ее символы и если счетчик ссылок принимает нулевое значение, то динамическая библиотека выгружается.  В случае успеха возвращает 0, иначе ненулевой результат.
3. **dlsym** – использует указатель на динамическую библиотеку, возвращаемую dlopen, и оканчивающееся нулем символьное имя, а затем возвращает адрес, указывающий на нужный символ. В случае неуспеха dlsym возвращает NULL.
4. **dlerror** – возвращает сообщение об ошибке, если ошибки не произошло, то возвращает NULL.
5. **Общий метод и алгоритм решения**

Программа принимает на вход неограниченное количество команд следующего вида:

* 0: переключение библиотеки. Выполняется при помощи функций dlopen, dlclose, dlsym, dlerror. Все системные ошибки обрабатываются.
* 1 x: расчет числа e по формуле или
* 2 num: перевод числа num из десятичной системы счисления в двоичную или троичную систему.

При вводе неверной команды программа выводит соответствующее сообщения. Для завершения программы можно использовать комбинацию Ctrl + D.

1. **Основные файлы программы**

**implementation.h**

#ifndef IMPLEMENTATION\_H

#define IMPLEMENTATION\_H

double e(int x);

extern char\* translation(long x);

#endif //IMPLEMENTATION\_H

**implementation-1.с**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

#include "implementation.h"

// Сounting 'e' number

double e(int x) {

    printf("Counting 'e' number using second remarkable limit\n");

    return pow(1 + 1. / x, x);

}

// Decimal to Binary Conversion

char\* translation(long x) {

    char \*res = calloc(1, sizeof(char));

    res[0] = '\0';

    if(x == 0) {

        char \*tmp = calloc(strlen(res) + 2, sizeof(char));

        strcpy(tmp+1, res);

        free(res);

        res = tmp;

        res[0] = '0';

    }

    while(x > 0) {

        char ch = (x & 1u) + '0';

        x >>= 1u;

        char \*tmp = calloc(strlen(res) + 2, sizeof(char));

        strcpy(tmp+1, res);

        free(res);

        res = tmp;

        res[0] = ch;

    }

    return res;

}

**implementation-2.с**

#include "implementation.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

// Counting 'e' number

double e(int x) {

    printf("Counting 'e' number using Taylor series\n");

    double ans = 0;

    long long factorial = 1;

    for (int n = 0; n <= x; ++n) {

        if (n != 0) {

            factorial \*= n;

        }

        ans += 1. / factorial;

    }

    return ans;

}

// Decimal to Ternary Conversion

char\* translation(long x) {

    char \*res = calloc(1, sizeof(char));

    res[0] = '\0';

    if(x == 0) {

        char \*tmp = calloc(strlen(res) + 2, sizeof(char));

        strcpy(tmp+1, res);

        free(res);

        res = tmp;

        res[0] = '0';

    }

    while(x > 0) {

        char ch = (x % 3) + '0';

        x /= 3;

        char \*tmp = calloc(strlen(res) + 2, sizeof(char));

        strcpy(tmp+1, res);

        free(res);

        res = tmp;

        res[0] = ch;

    }

    return res;

}

**programm-1.c**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "../lib/implementation.h"

void Menu() {

    puts("+-------------+------------------------------------------+");

    puts("|   command   |               description                |");

    puts("|=============+==========================================|");

    puts("|    1  x     |              count 'e' number            |");

    puts("|-------------+------------------------------------------|");

    puts("|    2  x     | translate number to binary number system |");

    puts("|             |        from decimal number system        |");

    puts("+-------------+------------------------------------------+\n");

}

int main() {

    Menu();

    int cmd;

    while (scanf("%d", &cmd) > 0) {

        if (cmd == 1) {

            int x;

            scanf("%d", &x);

            printf("%f\n", e(x));

        }

        else if (cmd == 2) {

            long x;

            if(scanf("%ld", &x)== EOF){

                break;

            }

            char\* returned\_str = translation(x);

            if (returned\_str == NULL) {

                perror("Memory allocation error.");

            }

            puts("Decimal to binary conversion");

            printf("%s\n", returned\_str);

            free(returned\_str);

        }

        else {

            puts("Incorrect command");

        }

    }

}

**Makefile**

# -fpic -- Position Independent Code\

           использовать относительную адресацию в переходах подпрограмм\

           во избежание конфликтов при динамическом связывании

# -shared -- предписывает создать динамическую библиотеку\

             именуйте динамические библиотеки следующим способом: libNAME.so

# -L. -- путь, по которму нужно искать файл библиотеки

# -lNAME\_OF\_LIBRARY -- линковка

# -lm -- для библиотеки math

# export LD\_LIBRARY\_PATH=. -- добавляем текущий каталог в $LD\_LIBRARY\_PATH

all: libimp.so

    gcc programm-1.c -L. -limp -Wall -o programm-1

libimp.so: libimp.o

    gcc libimp.o -lm -shared -o libimp.so

libimp.o:

    gcc ../lib/implementation-1.c -c -Wall -Werror -fpic -o libimp.o

clean:

    rm -rf \*.so \*.o programm-1

**programm-2.c**

#include <stdio.h>

#include <stdlib.h>

#include <dlfcn.h>

#include <unistd.h>

typedef enum {

    FIRST,

    SECOND,

} Context;

// default context for implementation

Context r = FIRST;

const char\* lib\_name\_1 = "libimp.so";

const char\* lib\_name\_2 = "libimp\_2.so";

void \*handler = NULL;

void loadDLibs(Context implementation){

    const char \*name;

    if(implementation == FIRST){

        name = lib\_name\_1;

    } else{

        name = lib\_name\_2;

    }

    // loading library

    // RTLD\_LAZY - perform lazy binding

    handler = dlopen(name, RTLD\_LAZY);

    if(!handler){

        fprintf(stderr, "%s\n", dlerror());

        exit(EXIT\_FAILURE);

    }

}

// resolving functions from library

double (\*e)(int) = NULL;

char\* (\*translation)(long x) = NULL;

char \*err;

void loadContext(){

    loadDLibs(r);

    // dslym() takes a "handle" of a dynamic library returned by dlopen()

    // and the null-terminated symbol name, returning the address where that symbol is loaded into memory

    e = dlsym(handler, "e");

    translation = dlsym(handler, "translation");

    if((err = dlerror())) {

        fprintf(stderr, "%s\n", err);

        exit(EXIT\_FAILURE);

    }

}

void changeContext(){

    dlclose(handler);

    if(r == FIRST){

        r = SECOND;

    } else {

        r = FIRST;

    }

    loadContext();

}

void Menu() {

    puts("+-------------+-------------------------------------------+");

    puts("|   command   |               description                 |");

    puts("|=============+===========================================|");

    puts("|      0      |           switch implementation           |");

    puts("|=========================================================|");

    puts("|              FISRT IMPLEMENTATION(default)              |");

    puts("|=========================================================|");

    puts("|     1  x    |               count 'e' number            |");

    puts("|             |        using second remarkable limit      |");

    puts("|-------------+-------------------------------------------|");

    puts("|    2 num    | translate number to binary number system  |");

    puts("|             |        from decimal number system         |");

    puts("|=========================================================|");

    puts("|                  SECOND IMPLEMENTATION                  |");

    puts("|=========================================================|");

    puts("|     1  x    |              count 'e' number             |");

    puts("|             |           using Taylor series             |");

    puts("|-------------+-------------------------------------------|");

    puts("|    2 num    | translate number to ternary number system |");

    puts("|             |        from decimal number system         |");

    puts("+-------------+-------------------------------------------+\n");

}

int main(){

    Menu();

    r = FIRST;

    loadContext();

    int cmd = 0;

    while (scanf("%d", &cmd) != EOF){

        if(cmd == 0){

            changeContext();

            puts("Changing context...");

            sleep(1);

            if(r == FIRST){

                puts("Now context is first");

            } else{

                puts("Now context is second");

            }

            continue;

        }

        if(cmd == 1) {

            int x;

            if (scanf("%d", &x) == EOF) {

                break;

            }

            printf("%f\n", e(x));

        }

        else if (cmd == 2) {

            long x;

            if(scanf("%ld", &x)== EOF){

                break;

            }

            char \*string;

            printf("Translate number %ld to ", x);

            if(r == FIRST) {

                printf("binary ");

            }else{

                printf("ternary ");

            }

            puts("number system");

            string = translation(x);

            printf("%s\n", string);

            free(string);

        }

        else {

            puts("Incorrect cmd");

        }

    }

    dlclose(handler);

}

**Makefile**

# -fpic -- Position Independent Code\

           использовать относительную адресацию в переходах подпрограмм\

           во избежание конфликтов при динамическом связывании

# -shared -- предписывает создать динамическую библиотеку\

             именуйте динамические библиотеки следующим способом: libNAME.so

# -L. -- путь, по которму нужно искать файл библиотеки

# -ldl -- для получения адреса функции нам потребуется dlsym — это функция из библиотеки libdl\

          которая найдет нашу функцию в стеке динамических библиотек

# -lm -- для библиотеки math

# export LD\_LIBRARY\_PATH=. -- добавляем текущий каталог в $LD\_LIBRARY\_PATH

all: libimp.so libimp\_2.so

    gcc programm-2.c -ldl -Wall -o programm-2

libimp.so: libimp.o

    gcc libimp.o -lm -shared -o libimp.so

libimp.o:

    gcc ../lib/implementation-1.c -c -Wall -Werror -fpic -o libimp.o

libimp\_2.so: libimp\_2.o

    gcc libimp\_2.o -shared -o libimp\_2.so

libimp\_2.o:

    gcc ../lib/implementation-2.c -c -Wall -Werror -fpic -o libimp\_2.o

clean:

    rm -rf \*.so \*.o programm-2

1. **Демонстрация работы программы**

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-1$** make

gcc ../lib/implementation-1.c -c -Wall -Werror -fpic -o libimp.o

gcc libimp.o -lm -shared -o libimp.so

gcc programm-1.c -L. -limp -Wall -o programm-1

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-1$** export LD\_LIBRARY\_PATH=.

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-1$** echo $LD\_LIBRARY\_PATH

.

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-1$** ./programm-1

+-------------+------------------------------------------+

| command | description |

|=============+==========================================|

| 1 x | count 'e' number |

|-------------+------------------------------------------|

| 2 x | translate number to binary number system |

| | from decimal number system |

+-------------+------------------------------------------+

1 10

Counting 'e' number using second remarkable limit

2.593742

1 100

Counting 'e' number using second remarkable limit

2.704814

2 8

Decimal to binary conversion

1000

2 512

Decimal to binary conversion

1000000000

0

Incorrect command

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-2$** make

gcc ../lib/implementation-1.c -c -Wall -Werror -fpic -o libimp.o

gcc libimp.o -lm -shared -o libimp.so

gcc ../lib/implementation-2.c -c -Wall -Werror -fpic -o libimp\_2.o

gcc libimp\_2.o -shared -o libimp\_2.so

gcc programm-2.c -ldl -Wall -o programm-2

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-2$** export LD\_LIBRARY\_PATH=.

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-2$** echo $LD\_LIBRARY\_PATH

.

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-2$** ./programm-2

+-------------+-------------------------------------------+

| command | description |

|=============+===========================================|

| 0 | switch implementation |

|=========================================================|

| FISRT IMPLEMENTATION(default) |

|=========================================================|

| 1 x | count 'e' number |

| | using second remarkable limit |

|-------------+-------------------------------------------|

| 2 num | translate number to binary number system |

| | from decimal number system |

|=========================================================|

| SECOND IMPLEMENTATION |

|=========================================================|

| 1 x | count 'e' number |

| | using Taylor series |

|-------------+-------------------------------------------|

| 2 num | translate number to ternary number system |

| | from decimal number system |

+-------------+-------------------------------------------+

1 10

Counting 'e' number using second remarkable limit

2.593742

1 10000

Counting 'e' number using second remarkable limit

2.718146

2 1024

Translate number 1024 to binary number system

10000000000

2 2048

Translate number 2048 to binary number system

100000000000

0

Changing context...

Now context is second

1 10

Counting 'e' number using Taylor series

2.718282

1 10000

Counting 'e' number using Taylor series

inf

2 1024

Translate number 1024 to ternary number system

1101221

2 2048

Translate number 2048 to ternary number system

2210212

0

Changing context...

Now context is first

1 10000

Counting 'e' number using second remarkable limit

2.718146

3

Incorrect command

1. **Strace**

Утилита, отслеживающая системные вызовы, которые являются механизмом трансляции, обеспечивающим взаимодействие между процессом и операционной системой. Эти вызовы могут быть перехвачены и прочитаны, что позволяет лучше понять, какую задачу процесс пытается выполнить в заданное время. Перехватывая эти вызовы, мы можем добиться лучшего понимания поведения процессов, особенно если что-то пошло не так. Команда ОС, позволяющая отслеживать системны вызовы, называется ptrace. Strace же вызывает ptrace и читает данные о поведении процесса, а затем выводит отчет.

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-1$** ltrace -o ltrace.log ./programm-1

+-------------+------------------------------------------+

| command | description |

|=============+==========================================|

| 1 x | count 'e' number |

|-------------+------------------------------------------|

| 2 x | translate number to binary number system |

| | from decimal number system |

+-------------+------------------------------------------+

1 10

Counting 'e' number using second remarkable limit

2.593742

2 8

Decimal to binary conversion

1000

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-1$** cat ltrace.log

puts("+-------------+-----------------"...) = 59

puts("| command | de"...) = 59

puts("|=============+================="...) = 59

puts("| 1 x | cou"...) = 59

puts("|-------------+-----------------"...) = 59

puts("| 2 x | translate number"...) = 59

puts("| | from deci"...) = 59

puts("+-------------+-----------------"...) = 60

\_\_isoc99\_scanf(0x7f2ce4600d54, 0x7fffc90d0524, 0x7f2ce3fdd8c0, 0x7fffc0aeb010) = 1

\_\_isoc99\_scanf(0x7f2ce4600d54, 0x7fffc90d0528, 0x7f2ce3fdd8d0, 16) = 1

e(10, 1, 0x7f2ce3fdd8d0, 16) = 0x4004bffc0c03023e

printf("%f\n", 2.593742) = 9

\_\_isoc99\_scanf(0x7f2ce4600d54, 0x7fffc90d0524, 0, 0) = 1

\_\_isoc99\_scanf(0x7f2ce4600d5b, 0x7fffc90d0528, 0x7f2ce3fdd8d0, 16) = 1

translation(8, 1, 0x7f2ce3fdd8d0, 16) = 0x7fffc0aebb00

puts("Decimal to binary conversion") = 29

puts("1000") = 5

free(0x7fffc0aebb00) = <void>

\_\_isoc99\_scanf(0x7f2ce4600d54, 0x7fffc90d0524, 0x7fffc0aeb010, 5) = 0xffffffff

+++ exited (status 0) +++

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-2$** ltrace -o ltrace.log ./programm-2

+-------------+-------------------------------------------+

| command | description |

|=============+===========================================|

| 0 | switch implementation |

|=========================================================|

| FISRT IMPLEMENTATION(default) |

|=========================================================|

| 1 x | count 'e' number |

| | using second remarkable limit |

|-------------+-------------------------------------------|

| 2 num | translate number to binary number system |

| | from decimal number system |

|=========================================================|

| SECOND IMPLEMENTATION |

|=========================================================|

| 1 x | count 'e' number |

| | using Taylor series |

|-------------+-------------------------------------------|

| 2 num | translate number to ternary number system |

| | from decimal number system |

+-------------+-------------------------------------------+

1 10

Counting 'e' number using second remarkable limit

2.593742

2 8

Translate number 8 to binary number system

1000

0

Changing context...

Now context is second

1 10

Counting 'e' number using Taylor series

2.718282

2 8

Translate number 8 to ternary number system

22

**ivan@Laptop-IM:/mnt/c/Users/Иван/projects/os\_labs/os\_lab5/task-2$** cat ltrace.log

puts("+-------------+-----------------"...) = 60

puts("| command | de"...) = 60

puts("|=============+================="...) = 60

puts("| 0 | switch"...) = 60

puts("|==============================="...) = 60

puts("| FISRT IMPLEMENTAT"...) = 60

puts("|==============================="...) = 60

puts("| 1 x | co"...) = 60

puts("| | using sec"...) = 60

puts("|-------------+-----------------"...) = 60

puts("| 2 num | translate number"...) = 60

puts("| | from deci"...) = 60

puts("|==============================="...) = 60

puts("| SECOND IMPLEM"...) = 60

puts("|==============================="...) = 60

puts("| 1 x | cou"...) = 60

puts("| | using "...) = 60

puts("|-------------+-----------------"...) = 60

puts("| 2 num | translate number"...) = 60

puts("| | from deci"...) = 60

puts("+-------------+-----------------"...) = 61

**dlopen("libimp.so", 1) = 0x7fffe4583690**

**dlsym(0x7fffe4583690, "e") = 0x7fbdf51e07ba**

**dlsym(0x7fffe4583690, "translation") = 0x7fbdf51e0810**

**dlerror() = nil**

\_\_isoc99\_scanf(0x7fbdf600136c, 0x7fffecb7b6b4, 1, 0) = 1

\_\_isoc99\_scanf(0x7fbdf600136c, 0x7fffecb7b6b8, 0x7fbdf57dd8d0, 16) = 1

printf("%f\n", 2.593742) = 9

\_\_isoc99\_scanf(0x7fbdf600136c, 0x7fffecb7b6b4, 0, 0) = 1

\_\_isoc99\_scanf(0x7fbdf6001373, 0x7fffecb7b6b8, 0x7fbdf57dd8d0, 16) = 1

printf("Translate number %ld to ", 8) = 22

printf("binary ") = 7

puts("number system") = 14

puts("1000") = 5

free(0x7fffe45847f0) = <void>

\_\_isoc99\_scanf(0x7fbdf600136c, 0x7fffecb7b6b4, 0x7fffe4583010, 5) = 1

**dlclose(0x7fffe4583690) = 0**

**dlopen("libimp\_2.so", 1) = 0x7fffe4583690**

**dlsym(0x7fffe4583690, "e") = 0x7fbdf51e074a**

**dlsym(0x7fffe4583690, "translation") = 0x7fbdf51e07c9**

**dlerror() = nil**

puts("Changing context...") = 20

sleep(1) = 0

puts("Now context is second") = 22

\_\_isoc99\_scanf(0x7fbdf600136c, 0x7fffecb7b6b4, 0x7fbdf57dd8c0, 3) = 1

\_\_isoc99\_scanf(0x7fbdf600136c, 0x7fffecb7b6b8, 0x7fbdf57dd8d0, 16) = 1

printf("%f\n", 2.718282) = 9

\_\_isoc99\_scanf(0x7fbdf600136c, 0x7fffecb7b6b4, 0, 0) = 1

\_\_isoc99\_scanf(0x7fbdf6001373, 0x7fffecb7b6b8, 0x7fbdf57dd8d0, 16) = 1

printf("Translate number %ld to ", 8) = 22

printf("ternary ") = 8

puts("number system") = 14

puts("22") = 3

free(0x7fffe4583ca0) = <void>

\_\_isoc99\_scanf(0x7fbdf600136c, 0x7fffecb7b6b4, 0, 0) = 0xffffffff

**dlclose(0x7fffe4583690) = 0**

+++ exited (status 0) +++

1. **Выводы**

В ходе данной лабораторной работы я познакомился с динамическими библиотеками в UNIX-подобной операционной системе. Чтобы разобраться с применением таких библиотек, мной были написаны две программы: одна подключала динамическую библиотеку на стадии линковки, а вторая – непосредственно во время исполнения.

В отличие от статических библиотек, динамические позволяют сделать зависящие от них приложения меньше по памяти за счет того, что их нужно лишь раз выгрузить в память, чтобы ей пользовались все, кто от нее зависит. Использование динамических библиотек существенно экономит память и ускоряет процесс сборки программы. Существует два способа использования динамических библиотек: динамическая компоновка в момент загрузки и динамическая загрузка на этапе исполнения. В первом случае всю работу по загрузке необходимых зависимостей выполняют операционная система и динамический компоновщик операционной системы. Операционная система Linux позволяет совершенствовать библиотеки «на ходу», когда программа уже запущена и использует какие-то библиотеки. Динамические библиотеки легче обновлять: достаточно исправить только код самой библиотеки.

Однако у динамических библиотек есть и свои недостатки. Например, это достаточно запутанная сборка программ. Также стоит отметить, что вызов функции из динамической библиотеки происходит немного медленнее.