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: Ильин И. О.

Группа: М8О–206Б–19

Вариант: 3

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

**Постановка задачи**

**Цель работы**

Целью является приобретение практических навыков в:

* Создание динамических библиотек
* Создание программ, которые используют функции динамических библиотек

**Задание**

Требуется создать динамические библиотеки, которые реализуют определенный 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 No1*), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
* Тестовая программа (*программа No2*), которая загружает библиотеки, используя только их местоположение и контракты.

Провести анализ двух типов использования библиотек.

Пользовательский ввод для обоих программ должен быть организован следующим образом:

1. Если пользователь вводит команду «0», то программа переключает одну реализацию

контрактов на другую (необходимо только для *программы No2*). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»;

1. «1 arg1 arg2 ... argN», где после «1» идут аргументы для первой функции, предусмотренной контрактами. После ввода команды происходит вызов первой функции, и на экране появляется результат её выполнения;
2. «2 arg1 arg2 ... argM», где после «2» идут аргументы для второй функции, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат её выполнения.

Вариант 33.

6. Рассчет значения числа е(основание натурального логарифма) *float e(int x)* как *(1 + 1/x) ^ x* и сумма ряда по *n* от *0* до *x*, где элементы ряда равны: *(1/(n!))*

9. Сортировка целочисленного массива *int\* sort(int\* array)* методами пузырьковой и быстрой сортировок.

**Общие сведения о программе**

Программы компилируются из двух файлов task1.c и task2.c. Программы используют динамические библиотеки libtools1.so и libtools2.so, которые компилируются из файлов my\_lib1.c, strconv.c и my\_lib2.c, stack.c соответственно. Использовались следующие заголовочные файлы: stdio.h, stdlib.h, stdbool.h, string.h, dlfcn.h., math.h, unistd.h, string.h, inttypes.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.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы компиляции для создания динамических библиотек, необходимые ключи
2. Написать общий Makefile для двух программ.
3. Изучить принципы работы dlsym, dlopen, dlclose.
4. Организовать простейший командный интерфейс в файлах test1.c и test2.c.
5. В файле test1.c подключить библиотеку на этапе компиляции.
6. В файле test2.c загрузить библиотечные функции в runtime, с помощью dlsym, dlopen, dlclose.

**Основные файлы программы**

**Makefile:**

CC = clang

CC\_FLAGS = -pedantic -Wall

LD\_FLAGS = -L.

CC\_LIB\_FLAGS = -c -pedantic -Wall -fPIC

LD\_LIB\_FLAGS = -shared

EXECUTABLE1 = task1

EXECUTABLE2 = task2

SRC\_EXE1 = task1.c

SRC\_EXE2 = task2.c

OBJ\_EXE1 = $(SRC\_EXE1:.c=.o)

OBJ\_EXE2 = $(SRC\_EXE2:.c=.o)

LIB1 = -ltools1

LIB2 = -ltools2

SRC\_LIB1 = ./libs/my\_lib1.c ./libs/strconv.c

SRC\_LIB2 = ./libs/my\_lib2.c ./libs/stack.c

ND\_SRC\_LIB1 = $(notdir $(SRC\_LIB1))

ND\_SRC\_LIB2 = $(notdir $(SRC\_LIB2))

OBJ\_LIB1 = $(ND\_SRC\_LIB1:.c=.o)

OBJ\_LIB2 = $(ND\_SRC\_LIB2:.c=.o)

SO\_LIB1 = libtools1.so

SO\_LIB2 = libtools2.so

all: $(EXECUTABLE1) $(EXECUTABLE2)

$(EXECUTABLE1) : $(SRC\_EXE1) $(SO\_LIB1)

$(CC) $(CC\_FLAGS) $(SRC\_EXE1) -o $(EXECUTABLE1) $(LD\_FLAGS) $(LIB1)

$(EXECUTABLE2) : $(SRC\_EXE2) $(SO\_LIB2)

$(CC) $(CC\_FLAGS) $(SRC\_EXE2) -o $(EXECUTABLE2) $(LD\_FLAGS) $(LIB2)

$(SO\_LIB1) : $(OBJ\_LIB1)

$(CC) $^ -o $@ $(LD\_LIB\_FLAGS)

$(OBJ\_LIB1) : $(SRC\_LIB1)

$(CC) $(CC\_LIB\_FLAGS) $^

$(SO\_LIB2) : $(OBJ\_LIB2)

$(CC) $^ -o $@ $(LD\_LIB\_FLAGS)

$(OBJ\_LIB2) : $(SRC\_LIB2)

$(CC) $(CC\_LIB\_FLAGS) $^

clean:

rm -f \*.o \*.so $(EXECUTABLE1) $(EXECUTABLE2)

**my\_lib1.h**

#pragma once

#include <math.h>

// Рассчет значения числа е(основание натурального логарифма)

float e(int x);

// Сортировка целочисленного массива

int\* sort(int\* array);

**my\_lib1.c**

#include "my\_lib1.h"

// (1 + 1/x) ^ x

float e(int x) {

return pow((1 + 1.0/(float)x), x);

}

void \_swap(int \*lhs, int \*rhs)

{

int tmp = \*lhs;

\*lhs = \*rhs;

\*rhs = tmp;

}

// Пузырек

int\* sort(int\* array) {

int n = array[0] ;

for (int i = 1; i < n - 1; ++i) {

for (int j = 1; j < n-i; ++j) {

if (array[j] > array[j+1]) {

\_swap(&array[j], &array[j+1]);

}

}

}

return array;

}

**my\_lib2.h**

#pragma once

// Рассчет значения числа е(основание натурального логарифма)

float e(int x);

// Сортировка целочисленного массива

int\* sort(int\* array);

**my\_lib2.c**

#include "my\_lib2.h"

#include "stack.h"

long long \_fact(int x) {

long long to\_return = 1;

for (int i = 1; i <= x; ++i) {

to\_return \*= i;

}

return to\_return;

}

// - Сумма ряда по n от 0 до x, где элементы ряда равны: (1/(n!))

float e(int x) {

float to\_return = 0;

for (int n = 0; n < x; ++n) {

to\_return += ( (float)1 / \_fact(n) );

}

return to\_return;

}

void \_swap(int \*lhs, int \*rhs)

{

int tmp = \*lhs;

\*lhs = \*rhs;

\*rhs = tmp;

}

int \_pick\_pivot(int\* a, int n)

{

//медиана из первого, последнего и серединного элементов, если элементов < 3 — береться последний

if (n <= 2)

return a[n-1];

int x = a[0];

int y = a[(n-1)/2];

int z = a[n-1];

if ((x - y) \* (z - x) >= 0) // (x >= y && x <= z) || (x <= y && x >= z)

return x;

else if ((y - x) \* (z - y) >= 0) // (y >= x && y <= z) || (y <= x && y >= z)

return y;

else

return z;

}

// Сортировка Хоара

int\* sort(int\* array) {

int\* a = &array[1];

int n = array[0] - 1;

stack s;

s\_init(&s);

\_arr ar = {a, n};

s\_push(&s, ar);

while (!s\_is\_empty(&s)) {

ar = s\_pop(&s);

a = ar.array;

n = ar.n;

if (n < 2)

continue;

int i = 0;

int j = n - 1;

double pivot = \_pick\_pivot(a, n);

do {

while (a[i] < pivot)

i++;

while (a[j] > pivot)

j--;

if (i <= j) {

if (a[i] > a[j])

\_swap(&a[i], &a[j]);

i++;

j--;

}

} while (i <= j);

ar.array = a;

ar.n = j + 1;

s\_push(&s, ar);

ar.array = a + i;

ar.n = n - i;

s\_push(&s, ar);

}

s\_destroy(&s);

return array;

}

**stack.h**

#pragma once

#include <stdbool.h>

#include <stddef.h>

#include <stdlib.h>

typedef struct stack stack;

typedef struct stack\_el stack\_el;

typedef struct \_arr \_arr;

struct \_arr{

int\* array;

int n;

};

struct stack\_el{

\_arr val;

stack\_el\* next;

};

struct stack{

stack\_el\* head;

};

void s\_init(stack\* s);

void s\_destroy(stack\* s);

bool s\_is\_empty(stack\* s);

\_arr s\_pop(stack\* s);

bool s\_push(stack\* s, \_arr val);

**stack.c**

#include "stack.h"

void s\_init(stack\* s){

s->head = NULL;

}

void s\_destroy(stack\* s){

while(!s\_is\_empty(s)){

s\_pop(s);

}

}

bool s\_is\_empty(stack\* s){

return s->head == NULL;

}

bool s\_push(stack\* s, \_arr val){

stack\_el \*element = (stack\_el\*)malloc(sizeof(stack\_el));

if (element == NULL){

return false;

}

element->next = s->head;

element->val = val;

s->head = element;

return true;

}

\_arr s\_pop(stack\* s){

stack\_el\* e = s->head;

\_arr v = e->val;

s->head = e->next;

free(e);

return v;

}

**strconv.h**

#pragma once

#include <stdio.h>

#include <unistd.h>

#include <stdbool.h>

#include <stdlib.h>

#define ERR\_MALLOC 3

#define ERR\_REALLOC 4

#define ERR\_READ 5

#define ERR\_WRITE 6

#define ERR\_INV\_INT 21

int read\_strings(char\*\*\* ptr\_buffer, size\_t\* res\_size); // Считывает строки через пробел до конца ввода

void str\_array\_to\_int(char\*\* strs, int nums[], size\_t n); // Преобразует массив строк в массив int'ов

**strconv.c**

#include "strconv.h"

int read\_strings(char\*\*\* ptr\_buffer, size\_t\* res\_size) { // Считывает строки через пробел до конца ввода

char c; // Текущий считываемый символ

int str\_idx = 0; // Индекс текущего слова

size\_t buffer\_size = 1; // Кол-во строк

char\*\* buffer = malloc(sizeof(char\*) \* buffer\_size);

if (!buffer) {

perror("Malloc error");

exit(ERR\_MALLOC);

}

int\* lengths = malloc(sizeof(int) \* buffer\_size);

if (!lengths) {

perror("Malloc error");

exit(ERR\_MALLOC);

}

lengths[0] = 0;

int str\_size = 1; // Длина текущей строки

buffer[0] = malloc(sizeof(char) \* str\_size);

if (!buffer[0]) {

perror("Malloc error");

exit(ERR\_MALLOC);

}

//size\_t read\_bytes = 0;

//while ((read\_bytes = read(fd, &c, sizeof(char))) > 0) {

bool is\_start = true;

int res\_scanf;

while ((res\_scanf = scanf("%c", &c)) == 1) {

// if ((read\_bytes != sizeof(char)) && (read\_bytes != 0)) {

// perror("Read error");

// exit(ERR\_READ);

// }

if (c == '\n') { // Конец ввода слов

break;

}

if (c == ' ') { // Следующее слово

if (is\_start) {

continue;

}

++str\_idx;

str\_size = 1;

continue;

}

is\_start = false;

// Увеличиваем буфер строк и длин

if (str\_idx >= buffer\_size) {

//printf("Increasing buffer\n");

++buffer\_size;

str\_size = 1;

buffer = realloc(buffer, sizeof(char\*) \* buffer\_size);

if (!buffer) {

perror("Realloc error");

exit(ERR\_REALLOC);

}

buffer[buffer\_size - 1] = malloc(sizeof(char) \* str\_size);

if (!buffer[buffer\_size - 1]) {

perror("Malloc error");

exit(ERR\_MALLOC);

}

for (size\_t i = 0; i < str\_size; ++i) {

buffer[buffer\_size - 1][i] = 0;

}

lengths = realloc(lengths, sizeof(int) \* buffer\_size);

if (!lengths) {

perror("Realloc error");

exit(ERR\_REALLOC);

}

lengths[buffer\_size - 1] = 0;

}

// Увеличиваем длину строки

if (lengths[str\_idx] == str\_size) {

//printf("Increasing str len\n");

str\_size += 16;

buffer[str\_idx] = realloc(buffer[str\_idx], sizeof(char) \* str\_size);

if (!buffer[str\_idx]) {

perror("Realloc error");

exit(ERR\_REALLOC);

}

}

buffer[str\_idx][lengths[str\_idx]] = c;

++lengths[str\_idx];

// printf(">>> Read '%c'\n", c);

// printf("buffer[str\_idx] = %s\n", buffer[str\_idx]);

// printf("str\_idx = %d\n", str\_idx);

// printf("buffer\_size = %zu\n", buffer\_size);

// printf("str\_size = %d\n", str\_size);

// printf("length[str\_idx] = %d\n",lengths[str\_idx]);

}

for (int i = 0; i < buffer\_size; ++i) {

buffer[i] = realloc(buffer[i], sizeof(char) \* lengths[i]);

if (!buffer[i]) {

perror("Realloc error");

exit(ERR\_REALLOC);

}

buffer[i][lengths[i]] = '\0';

}

\*res\_size = buffer\_size;

\*ptr\_buffer = buffer;

// printf("Result from read\_strings func:\n");

// for (int i = 0; i < \*res\_size; ++i) {

// printf("%d) \"%s\"\n", i, buffer[i]);

// }

return res\_scanf;

}

void str\_array\_to\_int(char\*\* strs, int nums[], size\_t n) { // Преобразует массив строк в массив int'ов

for (size\_t i = 0; i < n; ++i) {

///nums[i] = atoi(strs[i]);

char\* p = strs[i];

bool is\_negative = false;

int result = 0;

while(\*p != '\0') {

if (\*p == '-' || \*p == '+') {

if (p != strs[i]) {

perror("Invalid int");

exit(ERR\_INV\_INT);

}

if (\*p == '-') {

is\_negative = true;

}

} else if ('0' <= \*p && \*p <= '9') {

result = result \* 10 + (\*p - '0');

} else {

perror("Invalid int");

exit(ERR\_INV\_INT);

}

++p;

}

if (is\_negative) {

result \*= -1;

}

nums[i] = result;

}

}

**task1.c**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <unistd.h>

#include <inttypes.h>

#include <string.h>

#include <dlfcn.h>

#include "./libs/my\_lib1.h"

#include "./libs/strconv.h"

#define ERR\_INV\_ARGS 22

int main() {

int res\_scanf = 1;

char\*\* strs = NULL;

size\_t size = 0;

while (res\_scanf != EOF) {

printf("Enter commad\n");

res\_scanf = read\_strings(&strs, &size); //DL1

char\* cmd = strs[0];

if (!strcmp(cmd, "1")) {

if (size > 2) {

perror("Error: invalid args number\n");

exit(ERR\_INV\_ARGS);

}

char\* endptr = NULL;

int num = strtoimax(strs[1], &endptr, 10);

if (strcmp(endptr, "")) {

printf("Error: invalid int value = %s\n", strs[1]);

exit(ERR\_INV\_INT);

}

printf("e = %f\n", e(num));

} else if (!strcmp(cmd, "2")) {

int nums[size];

nums[0] = size;

str\_array\_to\_int(&strs[1], &nums[1], size-1); // DL1

sort(nums);

printf("arrray = {");

for (size\_t i = 1; i < size; i++) {

printf("%d, ", nums[i]);

}

printf("}\n");

} else if (res\_scanf != EOF) {

printf("Invalid cmd: \"%s\"\n", cmd);

}

}

return 0;

}

**task2.c**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <unistd.h>

#include <inttypes.h>

#include <string.h>

#include <dlfcn.h>

#define ERR\_INV\_INT 21

#define ERR\_INV\_ARGS 22

#define ERR\_DL\_CLOSE 30

#define ERR\_DL\_OPEN 31

#define ERR\_DL\_SYM 32

static const char\* DL1 = "libtools1.so";

static const char\* DL2 = "libtools2.so";

int dls\_open(void\*\* dl1\_handle, void\*\* dl2\_handle) {

\*dl1\_handle = dlopen(DL1, RTLD\_LAZY);

if (!\*dl1\_handle) {

perror(dlerror());

exit(ERR\_DL\_OPEN);

}

\*dl2\_handle = dlopen(DL2, RTLD\_LAZY);

if (!\*dl2\_handle) {

perror(dlerror());

exit(ERR\_DL\_OPEN);

}

return 0;

}

int dls\_close(void\*\* dl1\_handle, void\*\* dl2\_handle) {

int to\_return = 0;

if (dlclose(\*dl1\_handle) != 0) {

perror(dlerror());

to\_return = ERR\_DL\_CLOSE;

}

if (dlclose(\*dl2\_handle) != 0) {

perror(dlerror());

to\_return = ERR\_DL\_CLOSE;

}

if (to\_return != 0) {

exit(to\_return);

}

return to\_return;

}

int main() {

void\* dl1\_handle = NULL;

void\* dl2\_handle = NULL;

char\* error = NULL;

dls\_open(&dl1\_handle, &dl2\_handle);

int (\*read\_strings)(char\*\*\*, size\_t\*);

read\_strings = (int (\*)(char\*\*\*, size\_t\*))dlsym(dl1\_handle, "read\_strings");

error = dlerror();

if (error != NULL) {

perror(error);

return ERR\_DL\_SYM;

}

void (\*str\_array\_to\_int)(char\*\*, int\*, size\_t);

str\_array\_to\_int = (void (\*)(char\*\*, int\*, size\_t))dlsym(dl1\_handle, "str\_array\_to\_int");

error = dlerror();

if (error != NULL) {

perror(error);

return ERR\_DL\_SYM;

}

float (\*e)(int);

e = (float (\*)(int))dlsym(dl1\_handle, "e");

error = dlerror();

if (error != NULL) {

perror(error);

return ERR\_DL\_SYM;

}

int\* (\*sort)(int\*);

sort = (int\* (\*)(int\*))dlsym(dl1\_handle, "sort");

error = dlerror();

if (error != NULL) {

perror(error);

return ERR\_DL\_SYM;

}

int res\_scanf = 1;

char\*\* strs = NULL;

size\_t size = 0;

int dl\_num = 1;

printf("=== Using %s ===\n", DL1);

while (res\_scanf != EOF) {

printf("Enter commad\n");

res\_scanf = (\*read\_strings)(&strs, &size); //DL1

char\* cmd = strs[0];

if (!strcmp(cmd, "0")) {

if (dl\_num == 1) {

e = (float (\*)(int))dlsym(dl2\_handle, "e");

error = dlerror();

if (error != NULL) {

perror(error);

return ERR\_DL\_SYM;

}

sort = (int\* (\*)(int\*))dlsym(dl2\_handle, "sort");

error = dlerror();

if (error != NULL) {

perror(error);

return ERR\_DL\_SYM;

}

dl\_num = 2;

printf("=== Switched to %s ===\n", DL2);

} else {

e = (float (\*)(int))dlsym(dl1\_handle, "e");

error = dlerror();

if (error != NULL) {

perror(error);

return ERR\_DL\_SYM;

}

sort = (int\* (\*)(int\*))dlsym(dl1\_handle, "sort");

error = dlerror();

if (error != NULL) {

perror(error);

return ERR\_DL\_SYM;

}

dl\_num = 1;

printf("=== Switched to %s ===\n", DL1);

}

} else if (!strcmp(cmd, "1")) {

if (size > 2) {

perror("Error: invalid args number\n");

exit(ERR\_INV\_ARGS);

}

char\* endptr = NULL;

int num = strtoimax(strs[1], &endptr, 10);

if (strcmp(endptr, "")) {

printf("Error: invalid int value = %s\n", strs[1]);

exit(ERR\_INV\_INT);

}

printf("e = %f\n", (\*e)(num));

} else if (!strcmp(cmd, "2")) {

int nums[size];

nums[0] = size;

(\*str\_array\_to\_int)(&strs[1], &nums[1], size-1); // DL1

(\*sort)(nums);

printf("arrray = {");

for (size\_t i = 1; i < size; i++) {

printf("%d ", nums[i]);

}

printf("}\n");

} else if (res\_scanf != EOF) {

printf("Invalid cmd: \"%s\"\n", cmd);

}

}

return 0;

}

**Пример работы**

MacBook-Pro:src mr-ilin$ make

clang -c -pedantic -Wall -fPIC libs/my\_lib1.c libs/strconv.c

clang my\_lib1.o strconv.o -o libtools1.so -shared

clang -pedantic -Wall task1.c -o task1 -L. -ltools1

clang -c -pedantic -Wall -fPIC libs/my\_lib2.c libs/stack.c

clang my\_lib2.o stack.o -o libtools2.so -shared

clang -pedantic -Wall task2.c -o task2 -L. -ltools2

MacBook-Pro:src mr-ilin$ cat ../test/1\_t01.t

1 12

2 9 3 5 2 7 56

1 15

2 9 4 6 1 4 90 4 23 7

1 16

2 0 9 1 4 6 8 1 1

MacBook-Pro:src mr-ilin$ ./task1 < ../test/1\_t01.t

Enter commad

e = 2.613035

Enter commad

arrray = {2, 3, 5, 7, 9, 56, }

Enter commad

e = 2.632879

Enter commad

arrray = {1, 4, 4, 4, 6, 7, 9, 23, 90, }

Enter commad

e = 2.637928

Enter commad

arrray = {0, 1, 1, 1, 4, 6, 8, 9, }

Enter commad

MacBook-Pro:src mr-ilin$ cat ../test/1\_t02.t

1 10asf43

MacBook-Pro:src mr-ilin$ ./task1 < ../test/1\_t02.t

Enter commad

Error: invalid int value = 10asf43

MacBook-Pro:src mr-ilin$ cat ../test/1\_t03.t

2 9 9 9 9 9 9 9 9

2 1 9 1 9 1 9 1 9

2 9 8 7 6 5 1 2 3 4

MacBook-Pro:src mr-ilin$ ./task1 < ../test/1\_t03.t

Enter commad

arrray = {9, 9, 9, 9, 9, 9, 9, 9, }

Enter commad

arrray = {1, 1, 1, 1, 9, 9, 9, 9, }

Enter commad

arrray = {1, 2, 3, 4, 5, 6, 7, 8, 9, }

Enter commad

MacBook-Pro:src mr-ilin$ cat ../test/2\_t01.t

1 10

0

1 10

0

2 4 5 2 7 8 4 6 7

0

2 4 5 2 7 8 4 6 7

MacBook-Pro:src mr-ilin$ ./task2 < ../test/2\_t01.t

=== Using libtools1.so ===

Enter commad

e = 2.593742

Enter commad

=== Switched to libtools2.so ===

Enter commad

e = 2.718282

Enter commad

=== Switched to libtools1.so ===

Enter commad

arrray = {2 4 4 5 6 7 7 8 }

Enter commad

=== Switched to libtools2.so ===

Enter commad

arrray = {2 4 4 5 6 7 7 8 }

Enter commad

MacBook-Pro:src mr-ilin$ cat ../test/2\_t02.t

0

1 10asf43

MacBook-Pro:src mr-ilin$ ./task2 < ../test/2\_t02.t

=== Using libtools1.so ===

Enter commad

=== Switched to libtools2.so ===

Enter commad

Error: invalid int value = 10asf43

MacBook-Pro:src mr-ilin$ cat ../test/2\_t03.t

0

2 9 9 9 9 9 9 9 9

2 1 9 1 9 1 9 1 9

2 9 8 7 6 5 1 2 3 4

MacBook-Pro:src mr-ilin$ ./task2 < ../test/2\_t03.t

=== Using libtools1.so ===

Enter commad

=== Switched to libtools2.so ===

Enter commad

arrray = {9 9 9 9 9 9 9 9 }

Enter commad

arrray = {1 1 1 1 9 9 9 9 }

Enter commad

arrray = {1 2 3 4 5 6 7 8 9 }

Enter commad

**Вывод**

В ходе выполнения данной лабораторной работы я познакомился с тем, как создавать и использовать динамические библиотеки. Стоить отметить, что в простых программах с минимальной функциональностью использование статических библиотек может быть предпочтительней, однако в больших программах, использующих несколько библиотек, использование динамических библиотек позволяет снизить потребление оперативной и дисковой памяти, поскольку динамическую библиотеку достаточно один раз выгрузить в память, что ею могли пользоваться все нуждающиеся программы.

Существуют два способа использования динамических библиотек: динамическая компоновка и динамическая загрузка. Динамическая компоновка происходит в момент компиляции, загрузку ресурсов выполняет операционная система и ее динамических компоновщик. Динамическая загрузка происходит в *runtime* и предоставляет более широкие возможности: можно выбрать какие библиотеки подгружать, какие функции из них взять.