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: Киреев Александр Константинович

Группа: М8О–206Б–19

Вариант: 34

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 34:

7: Подсчет площади плоской геометрической фигуры по двум сторонам. Фигуры прямоугольник и прямоугольный треугольник.

8: Перевод числа x из десятичной системы счисления в другую. Другая система счисления – двоичная и троичная.

**Общие сведения о программе**

Программа компилируется из файлов lab5\_t1.c и lab5\_t2.c. Программа использует динамические библиотеки, которые компилируются из файлов square\_r.c, square\_t.c, translation\_b.c, translation\_t.c. Также используется заголовочные файлы: stdio.h, stdlib.h, stdint.h, stdbool.h, string.h, 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.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить ключи компилятора для создания динамических библиотек.
2. Изучить работу dlopen, dlclose, dlsym, dlerror.
3. Изучить принципы работы динамических библиотек, динамической линковки и динамической загрузки.
4. Реализовать функции для динамических библиотек.
5. Реализовать обработку не валидных данных и обработку ошибок в программе.
6. Реализовать make файлы.

**Основные файлы программы**

**lab5\_t1.c:**

void

skip\_str() {

int c = getchar();

while (c != '\n' && c != EOF) {

c = getchar();

}

}

void

handler\_loop() {

int32\_t cmd;

int8\_t ret\_scanf\_val = 0;

while (true) {

if ((ret\_scanf\_val = scanf("%d", &cmd)) == EOF) {

break;

} else if (ret\_scanf\_val == 0) {

printf("Invalid cmd, try again.\n");

skip\_str();

continue;

}

float a, b;

int64\_t num;

switch (cmd) {

case 1:

if ((ret\_scanf\_val = scanf("%f%f", &a, &b)) == EOF) {

break;

} else if (ret\_scanf\_val != 2) {

printf("Invalid args, try again.\n");

skip\_str();

break;

}

if (a <= 0.0 || b <= 0.0) {

printf("Sides must be positive, try again.\n");

break;

}

printf("Result is:\n%f\n", square(a, b));

break;

case 2:

if ((ret\_scanf\_val = scanf("%lld", &num)) == EOF) {

break;

} else if (ret\_scanf\_val == 0) {

printf("Invalid args, try again.\n");

skip\_str();

break;

}

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

if (returned\_str == NULL) {

perror("Memory allocation error.");

}

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

free(returned\_str);

break;

default:

printf("Invalid cmd, try again.\n");

break;

}

}

}

int

main() {

usage();

handler\_loop();

return 0;

}

**lab5\_t2.c:**

void

skip\_str() {

int c = getchar();

while (c != '\n' && c != EOF) {

c = getchar();

}

}

void

handler\_loop( void\* dl\_sqr\_handle, void\* dl\_trb\_handle,

void\* dl\_sqt\_handle, void\* dl\_trt\_handle) {

float (\*square)(float, float);

char\* (\*translation)(int64\_t);

char\* error = NULL;

square = (float (\*)(float, float))dlsym(dl\_sqr\_handle, SQ\_FUNC);

if ((error = dlerror()) != NULL) {

perror(error);

return;

}

translation = (char\* (\*)(int64\_t))dlsym(dl\_trb\_handle, TR\_FUNC);

if ((error = dlerror()) != NULL) {

perror(error);

return;

}

int32\_t cmd;

int8\_t ret\_scanf\_val = 0;

int8\_t realization = 0;

while (true) {

if ((ret\_scanf\_val = scanf("%d", &cmd)) == EOF) {

break;

} else if (ret\_scanf\_val == 0) {

printf("Invalid cmd, try again.\n");

skip\_str();

continue;

}

float a, b;

int64\_t num;

switch (cmd) {

case 0:

if (realization == 0) {

square = (float (\*)(float, float))dlsym(dl\_sqt\_handle, SQ\_FUNC);

if ((error = dlerror()) != NULL) {

perror(error);

return;

}

translation = (char\* (\*)(int64\_t))dlsym(dl\_trt\_handle, TR\_FUNC);

if ((error = dlerror()) != NULL) {

perror(error);

return;

}

} else {

square = (float (\*)(float, float))dlsym(dl\_sqr\_handle, SQ\_FUNC);

if ((error = dlerror()) != NULL) {

perror(error);

return;

}

translation = (char\* (\*)(int64\_t))dlsym(dl\_trb\_handle, TR\_FUNC);

}

printf("Switched realizations from %d to %d.\n", realization + 1, (realization + 1) % 2 + 1);

realization = (realization + 1) % 2;

break;

case 1:

if ((ret\_scanf\_val = scanf("%f%f", &a, &b)) == EOF) {

break;

} else if (ret\_scanf\_val != 2) {

printf("Invalid args, try again.\n");

skip\_str();

break;

}

if (a <= 0.0 || b <= 0.0) {

printf("Sides must be positive, try again.\n");

break;

}

printf("Result is:\n%f\n", (\*square)(a, b));

break;

case 2:

if ((ret\_scanf\_val = scanf("%lld", &num)) == EOF) {

break;

} else if (ret\_scanf\_val == 0) {

printf("Invalid args, try again.\n");

skip\_str();

break;

}

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

if (returned\_str == NULL) {

perror("Memory allocation error.");

}

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

free(returned\_str);

break;

default:

printf("Invalid cmd, try again.\n");

break;

}

}

}

int32\_t

dylb\_init( void\*\* dl\_sqr\_handle, void\*\* dl\_trb\_handle,

void\*\* dl\_sqt\_handle, void\*\* dl\_trt\_handle) {

\*dl\_sqr\_handle = dlopen(SQ\_R\_LIB, RTLD\_LAZY);

if (!(\*dl\_sqr\_handle)) {

perror(dlerror());

return SQR\_LIB\_CREATE\_ERR;

}

\*dl\_trb\_handle = dlopen(TR\_B\_LIB, RTLD\_LAZY);

if (!(\*dl\_trb\_handle)) {

if (dlclose((\*dl\_sqr\_handle)) != 0) {

fprintf(stderr, "Closing SQR lib error.");

}

perror(dlerror());

return TRB\_LIB\_CREATE\_ERR;

}

\*dl\_sqt\_handle = dlopen(SQ\_T\_LIB, RTLD\_LAZY);

if (!(\*dl\_sqt\_handle)) {

if (dlclose((\*dl\_sqr\_handle)) != 0) {

fprintf(stderr, "Closing SQR lib error.");

}

if (dlclose((\*dl\_trb\_handle)) != 0) {

fprintf(stderr, "Closing TRB lib error.");

}

perror(dlerror());

return SQT\_LIB\_CREATE\_ERR;

}

\*dl\_trt\_handle = dlopen(TR\_T\_LIB, RTLD\_LAZY);

if (!(\*dl\_trt\_handle)) {

if (dlclose((\*dl\_sqr\_handle)) != 0) {

fprintf(stderr, "Closing SQR lib error.");

}

if (dlclose((\*dl\_trb\_handle)) != 0) {

fprintf(stderr, "Closing TRB lib error.");

}

if (dlclose((\*dl\_sqt\_handle)) != 0) {

fprintf(stderr, "Closing SQT lib error.");

}

perror(dlerror());

return TRT\_LIB\_CREATE\_ERR;

}

return 0;

}

int32\_t

dylb\_close( void\*\* dl\_sqr\_handle, void\*\* dl\_trb\_handle,

void\*\* dl\_sqt\_handle, void\*\* dl\_trt\_handle) {

int32\_t to\_return = 0;

if (dlclose((\*dl\_sqr\_handle)) != 0) {

perror(dlerror());

to\_return = SQR\_LIB\_CLOSE\_ERR;

}

if (dlclose((\*dl\_trb\_handle)) != 0) {

perror(dlerror());

to\_return = TRB\_LIB\_CLOSE\_ERR;

}

if (dlclose((\*dl\_sqt\_handle)) != 0) {

perror(dlerror());

to\_return = SQT\_LIB\_CLOSE\_ERR;

}

if (dlclose((\*dl\_trt\_handle)) != 0) {

perror(dlerror());

to\_return = TRT\_LIB\_CLOSE\_ERR;

}

return to\_return;

}

int

main() {

void\* dl\_sqr\_handle = NULL;

void\* dl\_trb\_handle = NULL;

void\* dl\_sqt\_handle = NULL;

void\* dl\_trt\_handle = NULL;

int32\_t init\_return = 0;

if ((init\_return = dylb\_init(&dl\_sqr\_handle, &dl\_trb\_handle, &dl\_sqt\_handle, &dl\_trt\_handle)) != 0) {

return init\_return;

}

usage();

handler\_loop(dl\_sqr\_handle, dl\_trb\_handle, dl\_sqt\_handle, dl\_trt\_handle);

int32\_t close\_return = 0;

if ((close\_return = dylb\_close(&dl\_sqr\_handle, &dl\_trb\_handle, &dl\_sqt\_handle, &dl\_trt\_handle)) != 0) {

return close\_return;

}

return 0;

}

**Пример работы**

MacBook-Air-K:test AK$ ./wrapper.sh

[info] [2020-12-09 12:59:55] Stage #1.1 Compiling task1...

gcc -c -pedantic -Wall -Werror -fPIC ../lib/square\_r.c

gcc -o libsrt.so square\_r.o -shared

gcc -c -pedantic -Wall -Werror -fPIC ../lib/translation\_b.c

gcc -o libtrb.so translation\_b.o -shared

export DYLD\_LIBRARY\_PATH=.

gcc -pedantic -Wall -Werror lab5\_t1.c -o lab5\_t1 -L . -lsrt -ltrb

[info] [2020-12-09 12:59:56] Stage #1.2 Compiling task2...

gcc -c -pedantic -Wall -Werror -fPIC ../lib/square\_r.c

gcc -shared -o libsquarer.so square\_r.o

gcc -c -pedantic -Wall -Werror -fPIC ../lib/translation\_b.c

gcc -o libtranslationb.so translation\_b.o -shared

gcc -c -pedantic -Wall -Werror -fPIC ../lib/square\_t.c

gcc -o libsquaret.so square\_t.o -shared

gcc -c -pedantic -Wall -Werror -fPIC ../lib/translation\_t.c

gcc -o libtranslationt.so translation\_t.o -shared

export DYLD\_LIBRARY\_PATH=.

gcc -pedantic -Wall -Werror lab5\_t2.c -o lab5\_t2 -ldl

[info] [2020-12-09 12:59:57] Stage #2.1 Test generating for task1...

[info] [2020-12-09 12:59:58] Stage #2.2 Test generating for task2...

[info] [2020-12-09 12:59:58] Stage #3.1 Checking task1...

>>Checking test\_1.txt...

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

| command | description |

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

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

| 1 a b | calculate area of rectangle |

| | with sides a , b |

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

| 2 num | translate number to binary number system |

| | from decimal number system |

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

Result is:

8.170000

Result is:

54.339497

Result is:

0000000000000000000000000000000000000000000000000000110001000110

Result is:

40.859001

Result is:

0000000000000000000000000000000000000000000000000010010010001100

Result is:

10.200001

Result is:

14.289189

Result is:

1111111111111111111111111111111111111111111111111110101101010100

Result is:

1111111111111111111111111111111111111111111111111110010000001110

Result is:

54.898560

>>Checking test\_2.txt...

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

| command | description |

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

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

| 1 a b | calculate area of rectangle |

| | with sides a , b |

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

| 2 num | translate number to binary number system |

| | from decimal number system |

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

Invalid cmd, try again.

Sides must be positive, try again.

Invalid args, try again.

Invalid cmd, try again.

Invalid cmd, try again.

[info] [2020-12-09 12:59:58] Stage #3.2 Checking task2...

>>Checking test\_1.txt...

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

| command | description |

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

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

| 0 | switch realization |

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

|-------------| FISRT REALIZATION(default) |

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

| 1 a b | calculate area of rectangle |

| | with sides a , b |

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

| 2 num | translate number to binary number system |

| | from decimal number system |

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

|-------------| SECOND REALIZATION |

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

| 1 a b | calculate area of right triangle |

| | with sides a , b |

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

| 2 num | translate number to ternary number system |

| | from decimal number system |

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

Switched realizations from 1 to 2.

Result is:

19.087738

Result is:

12.492896

Result is:

22.940001

Result is:

31.289999

Result is:

12.372431

Result is:

27.106730

Result is:

11.053230

Switched realizations from 2 to 1.

Result is:

29.600000

Result is:

10.760400

Switched realizations from 1 to 2.

Result is:

20.341999

Result is:

34.335003

Switched realizations from 2 to 1.

>>Checking test\_2.txt...

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

| command | description |

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

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

| 0 | switch realization |

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

|-------------| FISRT REALIZATION(default) |

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

| 1 a b | calculate area of rectangle |

| | with sides a , b |

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

| 2 num | translate number to binary number system |

| | from decimal number system |

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

|-------------| SECOND REALIZATION |

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

| 1 a b | calculate area of right triangle |

| | with sides a , b |

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

| 2 num | translate number to ternary number system |

| | from decimal number system |

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

Invalid cmd, try again.

Sides must be positive, try again.

Invalid args, try again.

Invalid cmd, try again.

Invalid cmd, try again.

rm -f \*.o \*.so lab5\_t1

rm -f \*.o \*.so lab5\_t2

**Вывод**

Выполнив данную лабораторную работу, я познакомился с динамическими библиотеками. В отличии от статических, они позволяют сделать зависящие от них приложения меньше по памяти за счет того, что динамическую библиотеку нужно лишь раз выгрузить в память, чтобы ей пользовались все, кто от нее зависит. 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, а во втором случае мы сами можем явно в коде указать какие библиотеки и когда подгрузить, что дает нам большую гибкость в действиях.