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: Чистяков Д. М.

Группа: М8О–206Б–20

Вариант: 36

Преподаватель: Соколов Андрей Алексеевич

Оценка:

Дата: 24.03.22

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2022

**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 6.

1. Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e Float SinIntegral(float A, float B, float e) двумя способами: подсчет интеграла методом прямоугольников, подсчет интеграла методом трапеций.

7. Подсчет площади плоской геометрической фигуры по двум сторонам Float Square(float A, float B): фигура – прямоугольник, фигура - прямоугольный треугольник.

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.

**Основные файлы программы**

**Makefile:**

all: sprog.out dprog.out

sprog.out: libs1.a

gcc smain.c -L. -ls1 -Wall -o sprog.out -lm

libs1.a: lib1.o

ar rc libs1.a lib1.o

ranlib libs1.a

dprog.out: libd1.so libd2.so

gcc dmain.c -ldl -o dprog.out -lm

libd1.so: lib1.o

gcc lib1.o -lm -shared -o libd1.so -lm

libd2.so: lib2.o

gcc lib2.o -shared -o libd2.so -lm

lib1.o:

gcc lib1.c -c -Wall -Werror -fpic -o lib1.o -lm

lib2.o:

gcc lib2.c -c -Wall -Werror -fpic -o lib2.o -lm

clean:

rm -rf \*.a \*.so \*.o sprog.out dprog.out

**lib1.c:**

#include <stdlib.h>

char\* translation(long x)

{

char bin[256];

int len = 0;

if(x == 0)

{

bin[len] = '0';

len++;

}

while (x > 0)

{

bin[len] = '0' + (x % 2);

len++;

x /= 2;

}

char\* res = (char\*) malloc((len + 1) \* sizeof(char));

res[len] = '\0';

for(int i = len - 1; i >= 0; i--)

{

res[i] = bin[len - 1 - i];

}

return res;

}

int\* Sort(int\* array)

{

for (int i = 1; i < array[0] + 1; ++i)

{

for (int j = 1; j < array[0]; ++j)

{

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

{

int a = array[j];

array[j] = array[j + 1];

array[j + 1] = a;

}

}

}

return array;

}

**lib2.c:**

#include <stdlib.h>

char\* translation(long x)

{

char bin[256];

int len = 0;

if(x == 0)

{

bin[len] = '0';

len++;

}

while (x > 0)

{

bin[len] = '0' + (x % 3);

len++;

x /= 3;

}

char\* res = (char\*) malloc((len + 1) \* sizeof(char));

res[len] = '\0';

for(int i = len - 1; i >= 0; i--)

{

res[i] = bin[len - 1 - i];

}

return res;

}

static void \_\_sort(int\* a, int first, int last)

{

int i = first, j = last;

int tmp, x = a[(first + last) / 2];

do

{

while (a[i] < x)

{

i++;

}

while (a[j] > x)

{

j--;

}

if (i <= j)

{

if (i < j)

{

tmp = a[i];

a[i] = a[j];

a[j] = tmp;

}

i++;

j--;

}

} while (i <= j);

if (i < last)

{

\_\_sort(a, i, last);

}

if (first < j)

{

\_\_sort(a, first, j);

}

}

int\* Sort(int\* array)

{

\_\_sort(array, 1, array[0]);

return array;

}

**smain.c:**

#include <stdio.h>

#include <stdlib.h>

// Перевод числа x из десятичной системы счисления в другую

extern char\* translation(long x);

// Отсортировать целочисленный массив

extern int \* Sort(int \* array);

int main()

{

printf("1: translation\n");

printf("2: Sort\n");

int cmd = 0;

while (scanf("%d", &cmd) != EOF)

{

switch(cmd)

{

case 1:

printf("translation\n");

long a;

scanf("%ld", &a);

char\* str = translation(a);

printf("%s\n", str);

free(str);

break;

case 2:

printf("Sort\n");

int array[256];

int len = 0;

char c = '1';

int b;

while (c != '\n')

{

scanf("%d%c", &b, &c);

len++;

array[len] = b;

}

c = '1';

array[0] = len;

Sort(array);

for (int i = 1; i <= len; i++)

{

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

}

printf("\n");

break;

}

}

return 0;

}

**dmain.c:**

#include <stdio.h>

#include <stdlib.h>

#include <dlfcn.h>

int main()

{

printf("0: change lib\n");

printf("1: translation\n");

printf("2: Sort\n");

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

int\* (\*Sort) (int\* array) = NULL;

void \*handle1 = dlopen("./libd1.so", RTLD\_LAZY);

if (!handle1)

{

printf("%s\n", dlerror());

exit(1);

}

void \*handle2 = dlopen("./libd2.so", RTLD\_LAZY);

if (!handle2)

{

printf("%s\n", dlerror());

exit(2);

}

int lib = 1;

int cmd = 0;

char \*error;

while (scanf("%d", &cmd) != EOF)

{

switch(cmd)

{

case 0:

lib = 3 - lib;

printf("lib was changed to %d\n", lib);

break;

case 1:

if (lib == 1)

{

translation = dlsym(handle1, "translation");

}

else

{

translation = dlsym(handle2, "translation");

}

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

{

printf("dlsym error\n");

exit(3);

}

printf("translation\n");

long a;

scanf("%ld", &a);

char\* str = translation(a);

printf("%s\n", str);

free(str);

break;

case 2:

if (lib == 1)

{

Sort = dlsym(handle1, "Sort");

}

else

{

Sort = dlsym(handle2, "Sort");

}

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

{

printf("dlsym error\n");

exit(4);

}

printf("Sort\n");

int array[256];

int len = 0;

char c = '1';

int b;

while (c != '\n')

{

scanf("%d%c", &b, &c);

len++;

array[len] = b;

}

c = '1';

array[0] = len;

Sort(array);

for (int i = 1; i <= len; i++)

{

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

}

printf("\n");

break;

}

}

if (dlclose(handle1) != 0)

{

perror("dlclose error");

exit(5);

}

if (dlclose(handle2) != 0)

{

perror("dlclose error");

exit(6);

}

return 0;

}

**Пример работы**

den@den-VirtualBox:~/Документы/lab5$ make

gcc lib1.c -c -Wall -Werror -fpic -o lib1.o -lm

ar rc libs1.a lib1.o

ranlib libs1.a

gcc smain.c -L. -ls1 -Wall -o sprog.out -lm

gcc lib1.o -lm -shared -o libd1.so -lm

gcc lib2.c -c -Wall -Werror -fpic -o lib2.o -lm

gcc lib2.o -shared -o libd2.so -lm

gcc dmain.c -ldl -o dprog.out -lm

den@den-VirtualBox:~/Документы/lab5$ ./sprog.out

1: translation

2: Sort

1 128

translation

10000000

2 9 8 7 6 5 4 3 2 1

Sort

1 2 3 4 5 6 7 8 9

den@den-VirtualBox:~/Документы/lab5$ ./dprog.out

0: change lib

1: translation

2: Sort

0

lib was changed to 2

0

lib was changed to 1

1 1024

translation

10000000000

2 7 6 5 4

Sort

4 5 6 7

0

lib was changed to 2

1 256

translation

100111

1 27

translation

1000

2 6 5 9 7 1

Sort

1 5 6 7 9

**Вывод**

Выполняя данную лабораторную работу, я научился линковать статические библиотеки, а также подключать динамические библиотеки. Реализация модуля в виде статической библиотеки будет оптимально при небольшом объеме кода. Использование же динамических библиотеки позволит нам экономить память за счет того, что ей могут одновременно пользоваться несколько программ. Также с помощью динамических библиотек можно добиться полиморфизма на уровне модулей.