# Министерство науки и высшего образования РФ Федеральное государственное бюджетное образовательное учреждение

**высшего образования**

# «Московский Авиационный Институт» Национальный Исследовательский Университет

**Институт** №8 «Информационные технологии и прикладная математика»

**Кафедра** 806 «Вычислительная математика и программирование»

# Лабораторная работа №5

**по курсу «Операционные системы»**

|  |  |
| --- | --- |
| Студент: | Гуликов К.А. |
| Группа: | М8О-206Б-20 |
| Преподаватель: | Миронов Е. С. |
| Подпись: |  |
| Оценка: |  |
| Дата: |  |

Москва, 2022

# Содержание

1. Цель работы;
2. Постановка задачи;
3. Общие сведения о программе;
4. Общий метод и алгоритм решения;
5. Код программ;
6. Демонстрация работы программы;
7. Вывод.

# Цель работы

Приобретение практических навыков в:

* Создании динамических библиотек;
* Создании программ, которые используют функции динамических библиотек;

# Постановка задачи

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 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 знания полученные на этапе компиляции;
* Тестовая программа (программа №2), 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

№2). 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 8:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Описание | Сигнатура | Реализация1 | Реализация2 |
| 1 | Расчет интеграла sin(x) на отрезке [A; B] с шагом e | Float SinIntegral(float A, float B, float e) | Подсчет интеграла методом прямоугольников. | Подсчет интеграла методом трапеций. |
| 6 | Расчет значения числа е(основание натурального логарифма) | Float E(int x) | (1+1/x) ^ x | Сумма ряда по n от 0 до х, где элементы  ряда равны: (1/(n!)) |

**Общие сведения о программе**

Программа состоит из пяти файлов: program1.c, program2.c, lib1.c, lib2.c и Makefile. В данных файлах используются заголовочные файлы: stdio.h, stdlib.h, math.h, dlfcn.h.

# Общий метод и алгоритм решения

1. dlopen – загружает динамическую библиотеку;
2. dlclose – уменьшает счётчик ссылок на динамически загружаемый общий объект;
3. dlsym – использует указатель на динамическую библиотеку, возвращаемую dlopen, и оканчивающееся нулем символьное имя, а затем возвращает адрес, указывающий, откуда загружается этот символ. Если символ не найден, то возвращаемым значением dlsym является NULL;
4. dlerror – возвращает NULL, если не возникло ошибок с момента инициализации или его последнего вызова.

# Код программ

**program1.c:**

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

float SinIntegral(float A, float B, float e); float E(double);

int main(){

int com;

while(scanf("%d", &com) > 0) { if ( com == 1 ) {

printf("Enter line segment and step: "); float A,B,e;

if (scanf("%f%f%f", &A, &B, &e) != 3) { printf("Data entry error!\n");

}

printf("Result: %f\n", SinIntegral( A, B, e));

} else if (com == 2) { printf("Enter number: "); double h;

if (scanf("%lf", &h) != 1) { printf("Data entry error!\n");

}

printf("Result: %f\n", E(h));

}else{

printf("Wrong format\n");

}

}

}

# program2.c:

#include <stdio.h> #include <stdlib.h> #include <dlfcn.h>

int main(){

int num = 1;

void\* lib = dlopen("./lib1.so", RTLD\_LAZY); if (lib == NULL) {

printf("Failed to load dynamic link library!\n"); return 1;

}

float (\*SinIntegral) ( float, float, float ); float (\*E) (double);

\*(void\*\*) (&SinIntegral) = dlsym(lib, "SinIntegral");

\*(void\*\*) (&E) = dlsym(lib, "E"); char\* error = dlerror();

if (error != NULL) { printf("%s", error);

}

int com;

while(scanf("%d", &com) > 0) { if (com == 1) {

float A, B, e;

printf("Enter line segment and step: ");

if (scanf("%f%f%f", &A, &B, &e) != 3) { printf("Data entry error!\n");

}

printf("Result: %f\n", SinIntegral( A, B, e));

} else if (com == 2) { printf("Enter number: "); double h;

if (scanf("%lf", &h) != 1) { printf("Data entry error!\n");

}

printf("Result: %f\n", E(h));

} else if (com == 0) {

if (dlclose(lib) != 0) {

printf("Failed to upload dynamic library!\n");

}

if (num == 1) {

lib = dlopen("./lib2.so", RTLD\_LAZY); num = 2;

} else {

lib = dlopen("./lib1.so", RTLD\_LAZY); num = 1;

}

if (lib == NULL) {

printf("Failed to load dynamic library!\n"); return 1;

}

\*(void\*\*) (&SinIntegral) = dlsym(lib, "SinIntegral");

\*(void\*\*) (&E) = dlsym(lib, "E"); char\* error = dlerror();

if (error != NULL) { printf("%s", error);

}

} else {

printf("Wrong format\n");

}

}

}

# lib1.c:

#include <math.h>

float SinIntegral(float A, float B, float e){ double integral = 0;

for(float i = A; i < B; i+=e ){ integral += sin(i) \* e;

}

return integral;

}

float E(double x) { double k = 1 / x;

double e = pow((1 + k), x); return e;

}

# lib2.c:

#include <math.h>

float SinIntegral( float A, float B, float e){ double integral = 0;

for(float i = A + e; i < B; i+=e ){

integral += (sin(i) + sin(i - e)) / 2 \* e;

}

return integral;

}

int factorial(double count){ double num = 1;

for (double i = 1; i <= count; i++) num = num \* i;

return num;

}

float E(double x) { double s = 1;

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

s += (1 / k);

}

return s;

}

# Makefile:

FLAGS = -g -O2 -pedantic -std=c11 -Wall -Werror all: static1 static2 dynamic

static1: lib1 program1

gcc $(FLAGS) lib1.o program1.o -o stat1 -lm

lib1: lib1.c

gcc $(FLAGS) -c lib1.c -lm

static2: lib2 program1

gcc $(FLAGS) lib2.o program1.o -o stat2 -lm

lib2: lib2.c

gcc $(FLAGS) -c lib2.c -lm

program1: program1.c

gcc $(FLAGS) -c program1.c

dynamic: program2 dynamic1 dynamic2 gcc $(FLAGS) program2.o -o dyn3 -ldl

program2: program2.c

gcc $(FLAGS) -c program2.c -ldl

dynamic1: lib1.c

gcc $(FLAGS) -shared -o lib1.so -fPIC lib1.c -lm

dynamic2: lib2.c

gcc $(FLAGS) -shared -o lib2.so -fPIC lib2.c -lm clean:

rm -rf \*.o \*.so stat1 stat2 dyn3

# Демонстрация работы программы

1

Enter line segment and step: 0 3.14 0.1

Result: 1.999548

1

Enter line segment and step: 3.14 4.71 0.1

Result: -0.976689

2

Enter number: 5

Result: 2.488320

2

Enter number: 2

Result: 2.250000

0

1

Enter line segment and step: 0 3.14 0.1

Result: 1.997469

1

Enter line segment and step: 3.14 4.71 0.1

Result: -0.926899

2

Enter number: 5

Result: 2.716667

2

Enter number: 2

Result: 2.500000

# Вывод strace

konstantin@LAPTOP-44CRFC1U:~/labs/os/lab5$ ltrace -o ltrace.log ./solution1 1

Enter line segment and step: 0 3.14 0.1

Result: 1.999548

3

Enter number: 4

Result: 2.441406

konstantin@LAPTOP-44CRFC1U:~/labs/os/lab5$ cat ltrace.log

isoc99\_scanf(0x7fa0d4400d47, 0x7fffe70d8dc4, 0x7fffe70d8f08, 0x7fa0d4400c60) = 1

printf\_chk(1, 0x7fa0d4400cf0, 0x7fa0d3c4d8d0, 16) = 29

isoc99\_scanf(0x7fa0d4400d0e, 0x7fffe70d8dc8, 0x7fffe70d8dcc, 0x7fffe70d8dd0) = 3

|  |  |  |
| --- | --- | --- |
| sin(0, 3, 0x7fa0d3c4d8d0, 0x7fffe70d8873) |  | = 0 |
| sin(0, 3, 0, 0x7fffe70d8873) | = 0 |  |
| sin(0, 3, 0x3fb99999a0000000, 0x7fffe70d8873) |  | = 0 |
| sin(0, 3, 0x3fc99999a0000000, 0x7fffe70d8873) |  | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 154) | = 0 |  |
| sin(0, 3, 0x7fa0d3d721e0, 206) | = 0 |  |
| sin(0, 3, 0x7fa0d3d721e0, 258) | = 0 |  |
| sin(0, 3, 0x7fa0d3d721e0, 310) | = 0 |  |
| sin(0, 3, 0x7fa0d3d721e0, 362) | = 0 |  |
| sin(0, 3, 0x7fa0d3d721e0, 410) | = 0 |  |
| sin(0, 3, 0x7fa0d3d721e0, 346) | = 0 |  |
| sin(0, 3, 0x7fa0d3d721e0, 294) | = 0 |  |
| sin(0, 3, 0x7fa0d3d721e0, 242) | = 0 |  |
| sin(0, 3, 0x7fa0d3d721e0, 190) | = 0 |  |
| sin(0, 3, 0x7fa0d3d721e0, 142) | = 0 |  |

|  |  |
| --- | --- |
| sin(0, 3, 0x7fa0d3d721e0, 90) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 38) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 18) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 70) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 118) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 170) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 222) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 274) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 326) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 374) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 426) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 330) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 278) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 230) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 178) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 126) | = 0 |
| sin(0, 3, 0x7fa0d3d721e0, 74) | = 0 |

printf\_chk(1, 0x7fa0d4400d28, 0x4008cccc60000000, 74) = 17

isoc99\_scanf(0x7fa0d4400d47, 0x7fffe70d8dc4, 0x7fa0d3c4d8c0, 0) = 1

printf\_chk(1, 0x7fa0d4400d65, 0x7fa0d3c4d8d0, 16) = 14

isoc99\_scanf(0x7fa0d4400d74, 0x7fffe70d8dd0, 0x7fa0d3c4d8c0, 0) = 1

pow(0, 1, 0x7fa0d3c4d8d0, 0x7fffe70d8871) = 0

printf\_chk(1, 0x7fa0d4400d28, 0x4000000000000000, 561) = 17 lina\_tucha@LAPTOP-44CRFC1U:~/labs/os/lab5$ ltrace -o ltrace.log ./solution3 3

Enter number: 3

Result: 2.370370

0

3

Enter number: 3

Result: 2.666667

konstantin@LAPTOP-44CRFC1U:~/labs/os/lab5$ cat ltrace.log **dlopen("./lib1.so", 1) = 0x7fffe2214280 dlsym(0x7fffe2214280, "SinIntegral") = 0x7f021abe06f0 dlsym(0x7fffe2214280, "Sort") = 0x7f021abe0780**

**dlsym(0x7fffe2214280, "E") = 0x7f021abe07f0 dlerror() = nil**

isoc99\_scanf(0x7f021b800e5b, 0x7fffe944da44, 1, 0) = 1

printf\_chk(1, 0x7f021b800e77, 0x7f021b1dd8d0, 16) = 14

isoc99\_scanf(0x7f021b800e86, 0x7fffe944da50, 0x7f021b1dd8c0, 0) = 1

printf\_chk(1, 0x7f021b800e3c, 0x4000000000000000, 529) = 17

isoc99\_scanf(0x7f021b800e5b, 0x7fffe944da44, 0x7f021b1dd8c0, 0) = 1

**dlclose(0x7fffe2214280) = 0**

**dlopen("./lib2.so", 1) = 0x7fffe2214280 dlsym(0x7fffe2214280, "SinIntegral") = 0x7f021abe0740 dlsym(0x7fffe2214280, "Sort") = 0x7f021abe0900**

**dlsym(0x7fffe2214280, "E") = 0x7f021abe0950 dlerror() = nil**

isoc99\_scanf(0x7f021b800e5b, 0x7fffe944da44, 1, 0) = 1

printf\_chk(1, 0x7f021b800e77, 0x7f021b1dd8d0, 16) = 14

isoc99\_scanf(0x7f021b800e86, 0x7fffe944da50, 0x7f021b1dd8c0, 0) = 1

printf\_chk(1, 0x7f021b800e3c, 0x7f021b1dd8d0, 0x7fffe944d4d1) = 17

# Вывод

Выполнив данную лабораторную работу, я приобрёл практические навыки в создании динамических библиотек. Динамические библиотеки, хоть и замедляют загрузку программы, обладают существенными преимуществами перед статическими: нет необходимости копировать библиотеку для каждой отдельной программы, также в одном варианте возможно применять изменения библиотеки в уже запущенной программе. Существует два типа динамических библиотек. Первый из них предполагает линковку во время компиляции. При этом становится невозможно применять изменения, происходящие в библиотеке для запущенной программы. Эту проблему решает второй тип: библиотека, подгружаемая программой во время исполнения с помощью системных вызовов. Таким образом, использование динамических библиотек, подключаемых на этапе выполнения программы, является более гибким решением. Однако, при этом, нужно больше задумываться о безопасности.