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: Бутырев Даниил Вячеславович

Группа: М8О–206Б–20

Вариант: 7

Преподаватель: Соколов Андрей Алексеевич

Оценка:

Дата:

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Постановка задачи**

## Цель работы

Целью является приобретение практических навыков в:

1. Создание динамических библиотек
2. 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 №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»;
2. «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;
3. «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 | Реализация 1 | Реализация 2 |
| 1 | Рассчет интеграла функции 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. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | Перевод числа x из десятичной системы счисления в другую | Char\* translation(long x) | Другая система счисления двоичная | Другая система счисления троичная |

**Основные файлы программы**

**Makefile:**

# -rdynamic

# Pass the flag -export-dynamic to the ELF linker, on targets that support it. This instructs the linker to add all symbols, not only

# used ones, to the dynamic symbol table. This option is needed for some uses of "dlopen" or to allow obtaining backtraces from within a

# program.

# -static

# On systems that support dynamic linking, this overrides -pie and prevents linking with the shared libraries. On other systems, this

# option has no effect.

# -shared

# Produce a shared object which can then be linked with other objects to form an executable. Not all systems support this option. For

# predictable results, you must also specify the same set of options used for compilation (-fpic, -fPIC, or model suboptions) when you

# specify this linker option.[1]

#

#-fpic

# Generate position-independent code (PIC) suitable for use in a shared library, if supported for the target machine. Such code

# accesses all constant addresses through a global offset table (GOT). The dynamic loader resolves the GOT entries when the program

# starts (the dynamic loader is not part of GCC; it is part of the operating system). If the GOT size for the linked executable exceeds

# a machine-specific maximum size, you get an error message from the linker indicating that -fpic does not work; in that case, recompile

# with -fPIC instead. (These maximums are 8k on the SPARC, 28k on AArch64 and 32k on the m68k and RS/6000. The x86 has no such limit.)

# Position-independent code requires special support, and therefore works only on certain machines. For the x86, GCC supports PIC for

# System V but not for the Sun 386i. Code generated for the IBM RS/6000 is always position-independent.

# When this flag is set, the macros "\_\_pic\_\_" and "\_\_PIC\_\_" are defined to 1.

# -fPIC

# If supported for the target machine, emit position-independent code, suitable for dynamic linking and avoiding any limit on the size

# of the global offset table. This option makes a difference on AArch64, m68k, PowerPC and SPARC.

# Position-independent code requires special support, and therefore works only on certain machines.

# When this flag is set, the macros "\_\_pic\_\_" and "\_\_PIC\_\_" are defined to 2.

# -fpie

# -fPIE

# These options are similar to -fpic and -fPIC, but the generated position-independent code can be only linked into executables.

# Usually these options are used to compile code that will be linked using the -pie GCC option.

# -fpie and -fPIE both define the macros "\_\_pie\_\_" and "\_\_PIE\_\_". The macros have the value 1 for -fpie and 2 for -fPIE.

# -pie

# Produce a dynamically linked position independent executable on targets that support it. For predictable results, you must also

# specify the same set of options used for compilation (-fpie, -fPIE, or model suboptions) when you specify this linker option.

dll:

g++ -Wall -pedantic -fpic -shared e\_limit.cpp -o libe\_limit.so

g++ -Wall -pedantic -fpic -shared e\_summ.cpp -o libe\_summ.so

g++ -Wall -pedantic -fpic -shared sin\_rect.cpp -o libsin\_rect.so

g++ -Wall -pedantic -fpic -shared sin\_trap.cpp -o libsin\_trap.so

dp:

g++-11 -o prog\_dynamic\_t -fmodules-ts -pie -rdynamic explicit\_enum.cpp error\_string.cpp dynamic\_library.cpp prog\_dynamic\_time.cpp -ldl

sp:

g++-11 -o prog\_compile\_t -fmodules-ts -pie explicit\_enum.cpp prog\_compile\_time.cpp -L. -lsin\_trap -le\_summ -Wl,-rpath,.

clean:

rm \*.so prog\_dynamic\_t prog\_compile\_t

rm -r gcm.cache

**Остальное**

module;

#include <dlfcn.h>

#include <iostream>

#include <array>

import explicit\_enum;

import error\_string;

export module dynamic\_library;

namespace DynamicLibrary {

enum state {

first,

second,

};

state func\_type;

void\* sin\_rect;

void\* sin\_trap;

void\* e\_limit;

void\* e\_summ;

void check\_errors\_dlopen() {

char\* error;

if( ( error = dlerror() ) != ReturnValue::dlerror\_no\_error ) {

std::cerr << error << std::endl;

exit( ReturnValue::error\_dlopen\_fail );

}

}

void check\_errors\_dlsym() {

char\* error;

if( ( error = dlerror() ) != ReturnValue::dlerror\_no\_error ) {

std::cerr << error << std::endl;

exit( ReturnValue::error\_dlsym\_fail );

}

}

void check\_errors\_dlclose(int return\_val) {

if( return\_val != ReturnValue::dlclose\_success ) {

std::cerr << Error::dlclose\_fail() << std::endl;

exit( ReturnValue::error\_dlclose\_fail );

}

}

void first\_config( void\*& sin\_f, void\*& e\_f ) {

sin\_f = sin\_trap;

e\_f = e\_limit;

}

void second\_config( void\*& sin\_f, void\*& e\_f ) {

sin\_f = sin\_rect;

e\_f = e\_summ;

}

export void init( float ( \*&SinIntegral )( float, float, float ),

char\* ( \*&translation )( long ) ) {

sin\_trap = dlopen( "./libsin\_trap.so", RTLD\_LAZY );

check\_errors\_dlopen();

sin\_rect = dlopen( "./libsin\_rect.so", RTLD\_LAZY );

check\_errors\_dlopen();

e\_limit = dlopen( "./libe\_limit.so", RTLD\_LAZY );

check\_errors\_dlopen();

e\_summ = dlopen( "./libe\_summ.so", RTLD\_LAZY );

check\_errors\_dlopen();

void\* sin\_file\_ptr;

void\* e\_file\_ptr;

func\_type = first;

first\_config( sin\_file\_ptr, e\_file\_ptr );

SinIntegral = reinterpret\_cast< float( \* )( float, float, float ) >

( dlsym( sin\_file\_ptr, "SinIntegral" ) );

check\_errors\_dlsym();

translation = reinterpret\_cast< char\*( \* )( long ) >( dlsym( e\_file\_ptr, "translation" ) );

check\_errors\_dlsym();

}

export void switch\_realisation( float ( \*&SinIntegral )( float, float, float ),

char\* ( \*&translation )( long ) ) {

void\* sin\_file\_ptr;

void\* e\_file\_ptr;

switch( func\_type ) {

case first:

func\_type = second;

second\_config(sin\_file\_ptr, e\_file\_ptr);

break;

case second:

func\_type = first;

first\_config(sin\_file\_ptr, e\_file\_ptr);

break;

}

SinIntegral = reinterpret\_cast< float ( \* )( float, float, float ) >

( dlsym( sin\_file\_ptr, "SinIntegral" ) );

check\_errors\_dlsym();

translation = reinterpret\_cast< char\*( \* )( long ) >( dlsym( e\_file\_ptr, "translation" ) );

check\_errors\_dlsym();

}

export void detach() {

std::array file\_ptr{ sin\_rect, sin\_trap, e\_limit, e\_summ };

for( auto file : file\_ptr ) {

check\_errors\_dlclose( dlclose( file ) );

}

}

};

#include <vector>

// #include <cmath>

// extern "C"

// float E( int x ) {

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

// }

extern "C"

char\* translation( long x ) {

std::vector< char > abc;

bool negative;

if( x < 0 ) {

negative = true;

x \*= -1;

}

while( x > 0 ) {

abc.push\_back( '0' + x % 2 );

x /= 2;

}

char\* array = new char[ abc.size() ];

array[ abc.size() ] = '\0';

if( negative ) {

array[ 0 ] = '-';

array += 1;

}

for( std::vector<char>::size\_type i = 0; i < abc.size(); ++i ) {

array[ i ] = abc[ abc.size() - 1 - i ];

}

return negative ? --array : array;

}

#include <vector>

// #include <cmath>

// extern "C"

// float E( int x ) {

// float e\_val = 1.0f;

// float prev\_elem = 1.0f;

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

// prev\_elem = prev\_elem / i;

// e\_val += prev\_elem;

// }

// }

// return e\_val;

//3

extern "C"

char\* translation( long x ) {

std::vector< char > abc;

bool negative;

if( x < 0 ) {

negative = true;

x \*= -1;

}

while( x > 0 ) {

abc.push\_back( '0' + x % 3 );

x /= 3;

}

char\* array = new char[ abc.size() + 1 ];

array[ abc.size() ] = '\0';

if( negative ) {

array[ 0 ] = '-';

array += 1;

}

for( std::vector<char>::size\_type i = 0; i < abc.size(); ++i ) {

array[ i ] = abc[ abc.size() - 1 - i ];

}

return negative ? --array : array;

}

module;

#include <string>

export module error\_string;

export namespace Error {

std::string forking\_process( const int pid ) {

std::string return\_str = "error: forking process: ";

return return\_str + std::to\_string( pid );

}

std::string executing\_program( const std::string& pathname ) {

std::string return\_str = "error: executing program: ";

return return\_str + pathname;

}

std::string mmap\_failed( const int mapping\_length ) {

std::string return\_str = "error: mmap memory with length: ";

return return\_str + std::to\_string( mapping\_length );

}

std::string kill\_failed( pid\_t pid, int signal ) {

std::string return\_str = "error: kill failed : pid = ";

return return\_str + std::to\_string( static\_cast< int >( pid ) ) +

" and signal = " + std::to\_string( signal );

}

std::string opening\_file( const std::string& file ) {

std::string return\_str = "error: open failed, file: ";

return return\_str + file;

}

std::string writing\_to\_file( const int fd, const std::string& file ) {

std::string return\_str = "error: writing to file with fd: ";

return return\_str + std::to\_string( fd ) + " and sentence: " + file;

}

std::string dlclose\_fail() {

return "error: dlclose fail";

}

};

export module explicit\_enum;

export namespace StandardValue {

enum {

bits\_in\_byte = 8,

array\_begin = 0,

};

};

export namespace ReturnValue {

enum standard {

nice = 0,

};

enum exec {

bad\_exec = -1,

};

enum mmap {

mmap\_failed,

};

enum fork {

bad\_fork = -1,

fork\_child\_program = 0,

};

enum kill {

bad\_kill = -1,

};

enum open {

open\_fail = -1,

};

enum write {

write\_fail = -1,

};

const char\* dlerror\_no\_error = nullptr;

enum dl {

dlclose\_success = 0,

};

enum error {

error\_executing\_program = 1,

error\_forking\_process,

error\_opening\_file,

error\_not\_enough\_args,

error\_writing\_to\_file,

error\_dlclose\_fail,

error\_dlsym\_fail,

error\_dlopen\_fail,

};

};

export namespace MmapValue {

enum flags { //exactly

anon\_fd = -1,

anon\_offset = 0,

no\_offset = 0,

};

};

export namespace ExecValue {

enum args {

here\_not\_zero = 1,

here\_zero = 2,

not\_zero = 1,

};

extern const auto last\_arg = nullptr;

enum pointer {

program\_name = 0,

pointer\_arg = 1,

mask\_pointer\_arg = 2,

};

};

// enum {

// read\_end=0,

// write\_end=1,

// };

// enum {

// stdin = 0,

// stdout = 1,

// stderr = 2,

// };

// error\_creating\_pipe,

// error\_child\_args,

// error\_creating\_process,

// error\_reading\_input,

// error\_writing\_pipe,

// error\_reading\_pipe,

// error\_writing\_err,

// };

#include <iostream>

import explicit\_enum;

extern "C" {

float SinIntegral(float, float, float);

char\* translation(long);

}

int main(int argc, char \*argv[]) {

int command = 0;

const int count\_1\_func\_args = 3;

float args\_func1[ count\_1\_func\_args ];

const int count\_2\_func\_args = 1;

long args\_func2[ count\_2\_func\_args ];

while( true ) {

std::cin >> command;

if( std::cin.fail() )

return ReturnValue::nice;

switch( command ) {

case 1:

for( int i = 0; i < count\_1\_func\_args; ++i ) {

std::cin >> args\_func1[ i ];

if( std::cin.fail() )

return ReturnValue::nice;

}

std::cout << SinIntegral( args\_func1[ 0 ], args\_func1[ 1 ],

args\_func1[ 2 ] ) << std::endl;

break;

case 2:

for( int i = 0; i < count\_2\_func\_args; ++i ) {

std::cin >> args\_func2[ i ];

if( std::cin.fail() )

return ReturnValue::nice;

}

char\* delete\_this = translation( args\_func2[ 0 ] );

std::cout << delete\_this << std::endl;

delete delete\_this;

break;

}

}

return 0;

}

#include <iostream>

import dynamic\_library;

import explicit\_enum;

int main(int argc, char \*argv[]) {

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

// float (\*E)(int);

char\* (\*translation)(long);

int command = 0;

const int count\_1\_func\_args = 3;

float args\_func1[ count\_1\_func\_args ];

const int count\_2\_func\_args = 1;

long args\_func2[ count\_2\_func\_args ];

DynamicLibrary::init( SinIntegral, translation );

while( true ) {

std::cin >> command;

if( std::cin.fail() )

return ReturnValue::nice;

switch( command ) {

case 0:

DynamicLibrary::switch\_realisation( SinIntegral, translation );

break;

case 1:

for( int i = 0; i < count\_1\_func\_args; ++i ) {

std::cin >> args\_func1[ i ];

if( std::cin.fail() )

return ReturnValue::nice;

}

std::cout << SinIntegral( args\_func1[ 0 ], args\_func1[ 1 ],

args\_func1[ 2 ] ) << std::endl;

break;

case 2:

for( int i = 0; i < count\_2\_func\_args; ++i ) {

std::cin >> args\_func2[ i ];

if( std::cin.fail() )

return ReturnValue::nice;

}

char\* delete\_this = translation( args\_func2[ 0 ] );

std::cout << delete\_this << std::endl;

delete delete\_this;

break;

}

}

DynamicLibrary::detach();

return 0;

}

#include <cmath>

//#include <iostream>

#include <cstdio>

extern "C"

float SinIntegral( float A, float B, float e ) { //e - difference between steps

float integral = 0.0f;

if( std::signbit( B - A ) )

e \*= -1;

const int n = trunc( ( B - A ) / e );

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

integral += sin( A + ( e / 2.0f ) ) ;

A += e;

}

integral = integral \* e + sin( ( B + A ) / 2.0f ) \* ( B - A );

return integral;

}

#include <cmath>

#include <iostream>

#include <cstdio>

extern "C"

float SinIntegral( float A, float B, float e ) { //e - difference between steps

float integral = 0.0f;

if( std::signbit( B - A ) )

e \*= -1;

const int n = trunc( ( B - A ) / e );

if( n > 0 ) {

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

integral += ( sin( A ) + sin( A + e ) ) / 2.0f \* e;

A += e;

}

integral += ( sin( A ) + sin( B ) ) / 2.0f \* ( B - A );

// integral = sin( A ) / 4.0f; //cos I \*2 in last summ

// A += e;

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

// integral += sin( A );

// A += e;

// }

// integral = integral \* 2.0f \* e + sin( A ) \* ( B - A + e )

// + sin( B ) / 2.0f \* ( B - A );

} else {

integral = ( sin( A ) + sin( B ) ) / 2.0f \* ( B - A );

}

return integral;

}

**Пример работы**

steep@gg:~/educat/instit/os/5lab/src$ ./prog\_dynamic\_t

1 1 3 0.5

1.49828

2 10

1010

0

1 1 3 0.5

1.54635

2 10

101

2 100

10201

exit

steep@gg:~/educat/instit/os/5lab/src$ ./prog\_compile\_t

1 1 3 0.5

1.49828

2 10

101

2 100

10201

exit

**Вывод**

Можно легко использовать и создавать динамические библиотеки. Они оказываются полезными для больших проектов с многочисленными модулямми, для распространения программ и для использования кода разными программами.