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: Гаврилов Максим Сергеевич

Группа: М8О–206Б–20

Вариант: 7

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

**Постановка задачи**

## Цель работы

Приобретение практических навыков в:

* Управление процессами в ОС
* Обеспечение обмена данных между процессами посредством каналов

## 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.

**Общие сведения о программе**

Программа компилируется из файлов main.c, static.c, geometryLib1.c, geometryLib2.c,. Также используется заголовочные файлы: stdio.h, stdlib.h, string.h, math.h, unistd.h dlfcn.h. В программе используются следующие системные вызовы:

1. **dlopen** – загружает динамический общий объект (общую библиотеку) из файла, имя которого указано в строке filename (завершается null) и возвращает непрозрачный описатель на загруженный объект.
2. **dlsym** – функция возвращает адрес, по которому символ расположен в памяти (указывается одним из аргументов).
3. **dlclose** – уменьшает счётчик ссылок на динамически загружаемый общий объект, на который ссылается handle. Если счётчик ссылок достигает нуля, то объект выгружается. Все общие объекты, которые были автоматически загружены при вызове dlopen() для объекта, на

**Общий метод и алгоритм решения**.

1. Скомпилируем geometrylib1 и geometrylib2 с ключами -sharedи -fPIC, чтобы получить возможность подключать их как динамические библиотеки.
2. Чтобы использовать geometrylib1 в качестве статической библиотеки (для второй программы) создадим для нее заголовочный файл geometrylib1.h
3. В файле main.c создадим функции, которые, используя полный адрес динамических библиотек и имя необходимой нам функции будут будут доставать ее из библиотеки и производить вычисления.

**Основные файлы программы**

**Main.c**

|  |
| --- |
| //Гаврилов М.С. М8О-206Б-19 | Лр#5, вар. 7  #include <string.h>  #include <unistd.h>  #include <stdlib.h>  #include <math.h>  //#include "interface.h"  #include <stdio.h>  #include <dlfcn.h>  //#include "thread.h"  //#include "synch.h"  #define Pi 3.1415926  int exeFL\_t1 **(**char**\*** libname**,** char**\*** funcname**,** float**\*** funcArgs**,**float**\*** res**)** **{**  void**\*** lib**;**  char**\*** error**;**  lib **=** dlopen**(**libname**,** RTLD\_LAZY**);**  error **=** dlerror**();**  **if(**error **!=** **NULL)** **{**  printf**(**"%s\n"**,**error**);**  **return** **-**1**;**  **}**  float **(\***func**)(**float**,**float**,**float**);**  func **=** dlsym**(**lib**,**funcname**);**  error **=** dlerror**();**  **if(**error **!=** **NULL)** **{**  printf**(**"%s\n"**,**error**);**  **return** **-**2**;**  **}**  **\***res **=** **(\***func**)(**funcArgs**[**0**],**funcArgs**[**1**],**funcArgs**[**2**]);**    dlclose**(**lib**);**  **return** 0**;**  **}**  int exeFL\_t2 **(**char**\*** libname**,** char**\*** funcname**,** long**\*** funcArgs**,** char**\*\*** res**)** **{**  void**\*** lib**;**  char**\*** error**;**  lib **=** dlopen**(**libname**,** RTLD\_LAZY**);**  error **=** dlerror**();**  **if(**error **!=** **NULL)** **{**  printf**(**"%s\n"**,**error**);**  **return** **-**1**;**  **}**  char**\*** **(\***func**)(**long**);**  func **=** dlsym**(**lib**,**funcname**);**  error **=** dlerror**();**  **if(**error **!=** **NULL)** **{**  printf**(**"%s\n"**,**error**);**  **return** **-**2**;**  **}**  **\***res **=** **(\***func**)(**funcArgs**[**0**]);**    dlclose**(**lib**);**  **return** 0**;**  **}**  int main**()** **{**  char**\*** libPath**[**2**];**//заполнение адресов библиотек  libPath**[**0**]** **=** **(**char**\*)**malloc**(**200**);**  libPath**[**0**]** **=** strcpy**(**libPath**[**0**],**"/home/max/Рабочий стол/OS/lab5/prog/libOne.so"**);**  libPath**[**1**]** **=** **(**char**\*)**malloc**(**200**);**  libPath**[**1**]** **=** strcpy**(**libPath**[**1**],**"/home/max/Рабочий стол/OS/lab5/prog/libTwo.so"**);**  char**\*** funcNames**[**2**];**  funcNames**[**0**]** **=** **(**char**\*)**malloc**(**200**);**  funcNames**[**0**]** **=** strcpy**(**funcNames**[**0**],**"SinIntegral"**);**  funcNames**[**1**]** **=** **(**char**\*)**malloc**(**200**);**  funcNames**[**1**]** **=** strcpy**(**funcNames**[**1**],**"translation"**);**  int switcher **=** 0**,**cmdInd **=** 0**;**  **while(**scanf**(**"%d"**,&**cmdInd**)!=**EOF**)** **{**    **if(**cmdInd **==** 0**)** **{**  **if(**switcher **==** 0 **)** **{**  switcher **=** 1**;**  printf**(**"using lib two\n"**);**  **}** //переключение библиотек  **else** **{**  printf**(**"using lib one\n"**);**  switcher **=** 0**;**  **}**  **}**  **else**  **if(**cmdInd **==** 1**)** **{**  float args**[**3**];**  float Res**;**  scanf**(**"%f %f %f"**,&**args**[**0**],&**args**[**1**],&**args**[**2**]);**  **if** **(**exeFL\_t1**(**libPath**[**switcher**],**funcNames**[**cmdInd**-**1**],**args**,&**Res**)** **!=** 0**)** **{**  printf**(**"ERROR\n"**);**  **}**  **else** **{**  printf**(**"%lf\n"**,**Res**);**  **}**  **}**  **else**  **if(**cmdInd **==** 2**)** **{**  long args**[**1**];**  char**\*** Res**;**  scanf**(**"%li"**,&**args**[**0**]);**  **if** **(**exeFL\_t2**(**libPath**[**switcher**],**funcNames**[**cmdInd**-**1**],**args**,&**Res**)!=**0**)** **{**  printf**(**"ERROR\n"**);**  **}**  **else** **{**  printf**(**"%s\n"**,**Res**);**  **}**  **}**  **}**  **}** |

**Geometrylib1**

|  |
| --- |
| #include <stdlib.h>  #include <stdio.h>  #include <math.h>  #define prc 0.001  float SinIntegral**(**float A**,**float B**,**float d**)** **{**  float sum **=** 0**;**  **for(**float i **=** A**;**i **<=** B**;** i **+=** d**)** **{**  sum**+=** sin**(**i**)\***d**;**  **}**  **return** sum**;**  **}**  char**\*** translation**(**long A**)** **{**  int i**=**0**;**  int s **=** 10**;**  char**\*** res **=** **(**char**\*)**malloc**(**s**);**  **while(**A**>**0**)** **{**  **if(**i **>=** s**)** **{**  s**\*=**2**;**  res **=** **(**char**\*)**realloc**(**res**,**s**);**  **}**  **if(**A**%**2 **==** 1**)** **{**  res**[**i**]** **=** '1'**;**  **}**  **else** **{**  res**[**i**]** **=** '0'**;**  **}**  A**/=**2**;**  **++**i**;**  **}**  res**[**i**]** **=** 0**;**  i**--;**  char**\*** str **=** **(**char**\*)**malloc**(**s**);**  int j **=** 0**;**  **for(**j **=** 0**;**j**<=**i**;++**j**)** **{**  str**[**j**]** **=** res**[**i**-**j**];**  **}**  str**[**j**]** **=** 0**;**  free**(**res**);**  **return** str**;**  **}** |

**Geometrylib2**

|  |
| --- |
| #include <stdlib.h>  #include <math.h>  float SinIntegral**(**float A**,**float B**,**float d**)** **{**  float sum **=** 0**;**  **for(**float i **=** A**;**i **<=** B**;** i **+=** d**)** **{**  sum**+=** **((**sin**(**i**)+**sin**(**i**+**d**))/**2**)\***d**;**  **}**  **return** sum**;**  **}**  char**\*** translation**(**long A**)** **{**  int i**=**0**;**  int s **=** 10**;**  char**\*** res **=** **(**char**\*)**malloc**(**s**);**  **while(**A**>**0**)** **{**  **if(**i **>=** s**)** **{**  s**\*=**2**;**  res **=** **(**char**\*)**realloc**(**res**,**s**);**  **}**  **if(**A**%**3 **==** 2**)** **{**  res**[**i**]** **=** '2'**;**  **}**  **else**  **if(**A**%**3 **==** 1**)** **{**  res**[**i**]** **=** '1'**;**  **}**  **else** **{**  res**[**i**]** **=** '0'**;**  **}**  A**/=**3**;**  **++**i**;**  **}**  res**[**i**]** **=** 0**;**  i**--;**  char**\*** str **=** **(**char**\*)**malloc**(**s**);**  int j **=** 0**;**  **for(**j **=** 0**;**j**<=**i**;++**j**)** **{**  str**[**j**]** **=** res**[**i**-**j**];**  **}**  str**[**j**]** **=** 0**;**  free**(**res**);**  **return** str**;**  **}** |

**Static.c**

|  |
| --- |
| //Гаврилов М.С. М8О-206Б-19 | Лр#5, вар. 21  #include <string.h>  #include <unistd.h>  #include <stdlib.h>  //#include "interface.h"  #include <stdio.h>  #include "geometryLib1.h"  //#include "thread.h"  //#include "synch.h"  int main**()** **{**  int cmdInd **=** 0**;**  **while(**scanf**(**"%d"**,&**cmdInd**)!=**EOF**)** **{**  **if(**cmdInd **==** 1**)** **{**  float args**[**3**];**  float Res**;**  // float (\*func)(float,float,float);  scanf**(**"%f %f %f"**,&**args**[**0**],&**args**[**1**],&**args**[**2**]);**  Res **=** SinIntegral**(**args**[**0**],**args**[**1**],**args**[**2**]);**  printf**(**"%lf\n"**,**Res**);**  **}**  **else**  **if(**cmdInd **==** 2**)** **{**  long args**[**1**];**  char**\*** Res**;**  // char (\*func)(long);  scanf**(**"%li"**,&**args**[**0**]);**  Res **=** translation**(**args**[**0**]);**  printf**(**"%s\n"**,**Res**);**  **}**  **}**  **}** |

**Пример работы**

max@max-Swift:~/Рабочий стол/OS/lab5/prog$ ./ProgOne.out

1 1 2 0.1

0.952261

2 12

1100

2 65

1000001

0

1 1 2 0.1

0.952261

2 12

1100

2 65

1000001

0

q

^C

max@max-Swift:~/Рабочий стол/OS/lab5/prog$ ./ProgTwo.out

1 1 2 0.1

0.952261

2 12

1100

2 65

1000001

0

using lib two

1 1 2 0.1

0.955652

2 12

110

2 65

2102

^C

max@max-Swift:~/Рабочий стол/OS/lab5/prog$ max@max-Swift:~/Рабочий стол/OS/lab5/prog$ ./ProgOne.out

1 1 2 0.1

0.952261

2 12

1100

2 65

1000001

0

1 1 2 0.1

0.952261

2 12

1100

2 65

1000001

0

q

^C

max@max-Swift:~/Рабочий стол/OS/lab5/prog$ ./ProgTwo.out

1 1 2 0.1

0.952261

2 12

1100

2 65

1000001

0

using lib two

1 1 2 0.1

0.955652

2 12

110

2 65

2102

^C

max@max-Swift:~/Рабочий стол/OS/lab5/prog$

**Вывод**

Динамические библиотеки позволяют создавать программы, занимающие куда меньше памяти, что, несомненно, очень полезно в больших и сложных проектах. Научившись работать с ними, я приобрел навык, который, скорее всего, еще понадобиться мне в будущем.