**Федеральное государственное автономное образовательное учреждение**

**Высшего профессионального образования**

Санкт-Петербургский политехнический университет

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №5**

«Создание статических и динамических библиотек (Linux, Windows)»

Работу выполнил студент гр. № 53501/3 Цыганов А.А.

Работу принял преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Душутина Е.В.

Санкт-Петербург

2016

# Цель работы:

# На примере простого, но функционально полезного (с точки зрения системного применения) кода разработать шаблонные решения разработки библиотечных функций и их объединения в статическую и динамическую библиотеки для двух ОС. Таким образом, чтобы эти решения применить в дальнейшем для организации соответствующих библиотек функций обработки событий – «ловушек» в Windows и их аналогов в Linux.

# Ход работы:

Для работы была использована виртуальная машина VMware Workstation 10.0.4, с образом ОС Linux Ubuntu 64-bit и Visual Studio 2015 в ОС Windows 10 Pro.В ходе работыбудут рассмотрены примеры создания разделяемых библиотек (статические и динамические), и реализованы функции для определения размера файла и вывод информации о установленных жестких дисках: название, общий объем, объем свободного и занятого места.

# Разработка статических библиотек в Windows

Статическая библиотека — файл с исходным кодом или [объектный файл](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C), предназначенный для вставки в [программу](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D0%BD%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB) на этапе компоновки, содержащая набор исполняемых функций. Библиотеки, распространяемые в виде исходного кода, преобразуются [компилятором](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в [объектные файлы](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C). Затем [компоновщик](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%BE%D0%B2%D1%89%D0%B8%D0%BA) соединяет объектные файлы библиотек и объектные файлы исходной программы в один исполняемый файл. Компоновщик переносит исполняемый код из библиотеки в результирующий .exe-файл, в результате чего программа получает возможность использовать функции библиотеки.

Создадим проект Visual Studio 2015, использующий шаблон Win32 Console Application (тип Static library). Далее реализуем класс MyInfoHelper с двумя методами: fileSize и totalHDInfo. Для получения размера файла используем функцию WinAPI GetFileSize, и для информации о жестких дисках будут использоваться функции GetLogicalDriveStrings и GetDiskFreeSpaceEx. Ниже приведен код решения для статической библиотеки (состоит из 2х файлов):

*Файл MyInfoHelper.h*

#include <windows.h>

namespace MyInfoHelper

{

class MyInfoHelper

{

public:

static int fileSize(wchar\_t\* path);

static void totalHDInfo();

};

}

*Файл MyInfoHelper.cpp*

#include "MyInfoHelper.h"

#include <tchar.h>

#include <stdio.h>

#include <iostream>

#include <strsafe.h>

using namespace std;

namespace MyInfoHelper

{

float convertToGB(\_\_int64 size)

{

return size / (1024 \* 1024 \* 1024);

}

int MyInfoHelper::fileSize(wchar\_t\* path) {

HANDLE hFile = CreateFile(path,

GENERIC\_READ, // open for reading

0, // do not share

NULL, // default security

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL, // normal file

NULL);

DWORD size = GetFileSize(hFile, NULL);

CloseHandle(hFile);

if (size == INVALID\_FILE\_SIZE && (size = GetLastError()) != NO\_ERROR)

return -1;

else

return size;

}

void MyInfoHelper::totalHDInfo() {

cout << "Total HD info: : " << endl;

int sz = GetLogicalDriveStrings(NULL, 0);

TCHAR\* szLogicalDrives = new TCHAR[sz];

GetLogicalDriveStrings(sz, szLogicalDrives);

while (\*szLogicalDrives)

{

TCHAR szDisk[80];

lstrcpy(szDisk, szLogicalDrives);

UINT uDriveType = GetDriveType(szDisk);

if (uDriveType == DRIVE\_FIXED)

{

wcout << szDisk << endl;

\_\_int64 free=0, total=0;

GetDiskFreeSpaceEx(szDisk, NULL, (PULARGE\_INTEGER)&total, (PULARGE\_INTEGER)&free);

cout << "Total : " << convertToGB(total) << " GB" << endl;

cout << "Free : " << convertToGB(free) << " GB" << endl;

cout << "Used : " << convertToGB(total - free) << " GB" << endl;

}

while (\*szLogicalDrives) szLogicalDrives++;

szLogicalDrives++;

}

}

}

В классе MyInfoHelper реализованы две статические функции: totalHDInfo и fileSize. Полученные проект статической библиотеки может быть собран в конечный файл LabWindowsStaticLib.lib, который может использоваться в других проектах.

Далее создадим консольное приложение, которое будет вызывать соответствующие функции. Для этого Visual Studio позволяет добавить новый проект в уже существующее решение. Таким образом в решении будет два проекта: LabWindowsStaticLib и TestLabWindowsStaticLib. Ниже приведен код второй консольной программы, которая будет вызывать библиотечные функции из класса MyInfoHelper:

#include "stdafx.h"

#include "MyInfoHelper.h"

#include <iostream>

using namespace std;

int main()

{

wchar\_t\* path = L"C:\\vm\\OS X Yosemite\\OS X Yosemite.vmdk";

int size = MyInfoHelper::MyInfoHelper::fileSize(path);

printf("MyInfoHelper\n");

wcout << "File size of " << path << " - " << size << endl;

MyInfoHelper::MyInfoHelper::totalHDInfo();

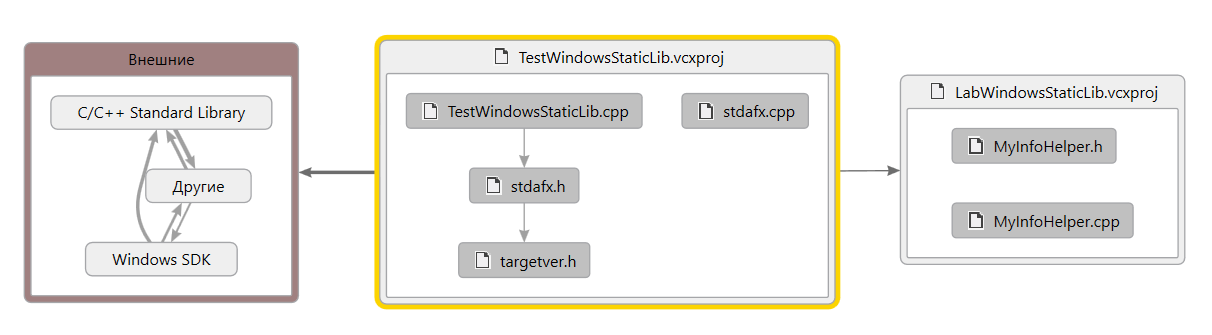
system("pause");

return 0;

}

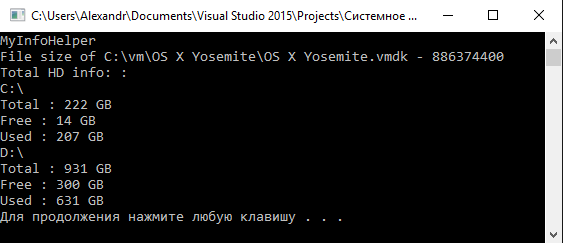
Для того что бы использовать статическую библиотеку в консольном приложении, необходимо в Visual Studio установить соответствующую связь между TestLabWindowsStaticLib и LabWindowsStaticLib.

Так же при сборке консольного приложения необходимо указать заголовочный файл на статическую библиотеку. После выполнения этих действий можно собирать решение целиком и запускать консольное приложение. Ниже показана связь между проектами.

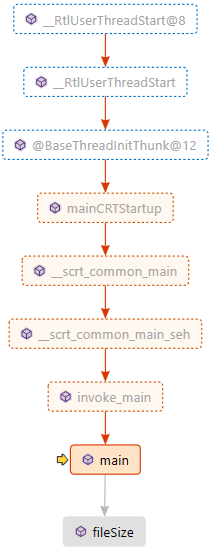


Результат сборки – объектный файл static.lib и исполняемый файл console.exe, содержащий статическую библиотеку.

После выполнения программы были получены следующие результаты:



Ниже приведен stack trace на момент выполнения функции fileSize из статической библиотеки:



# Создание динамических библиотек в Windows

Динамическая библиотека — файл, содержащий [машинный код](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4). Загружается в [память](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) [процесса](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) [загрузчиком программ](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D1%87%D0%B8%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC) [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) либо при создании процесса, либо по запросу [уже работающего процесса](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%BD%D1%82%D0%B0%D0%B9%D0%BC), то есть динамически. Динамические библиотеки подключаются к приложению во время его запуска или по ходу работы (run time).

Создадим динамическую библиотеку, которая будет содержать такой же класс и функции как в статической библиотеке.

Можно выделить следующие преимущества динамических библиотек над статическими:

* Экономия памяти, за счёт использования одной копии библиотеки несколькими исполняемыми файлами;
* При измении\редактировании библиотеки, исполняемый файл не пересобирая. При этом библиотека должна строго сохранять неизменным программный (API) и бинарный (ABI) интерфейс, иначе могут возникнуть ошибки при запуске и работе приложения, использующего библиотеку.

К основным недостаткам динамических библиотек можно отнести: процесс отладки приложения, например, на этапе линковки; сложность распространения приложений, использующих динамические библиотеки. В пакет с приложением нужно либо вручную добавлять все используемые библиотеки, либо указывать точную версию библиотеки, с которой предполагается работа.

В Visual Studio предоставляет следующие методы использования динамических библиотек:

1. Указать путь к библиотеке компоновщику. Никаких изменений в коде приложения не потребуется, подключение библиотеки будет происходить при запуске исполняемого файла.
2. Вручную подключить библиотеку с помощью функции LoadLibrary. Подключение будет происходить при вызове функции.

Воспользуемся существующим решением с статической библиотекой и консольным приложением и добавим новый проект с динамической библиотекой. Ниже приведено содержание класса MyInfoHelperD с соответствующими функциями.

*Файл MyInfoHelperD.h*

#pragma once

#ifdef MATHFUNCSDLL\_EXPORTS

#define MATHFUNCSDLL\_API \_\_declspec(dllexport)

#else

#define MATHFUNCSDLL\_API \_\_declspec(dllimport)

#endif

namespace MyInfoHelperD

{

class MyInfoHelperD

{

public:

static MATHFUNCSDLL\_API int fileSize(wchar\_t\* path);

static MATHFUNCSDLL\_API void totalHDInfo();

};

}

Когда символ MATHFUNCSDLL\_EXPORTS определен, символ MATHFUNCSDLL\_API установит модификатор\_\_declspec(dllexport) в объявлении функции класса. Этот модификатор разрешает экспорт функции библиотеки DLL для использования ее другими приложениями. Если символ MATHFUNCSDLL\_EXPORTS не определен, например, когда заголовочный файл включен приложением, символ LIBDLL \_API определяет модификатор \_\_declspec(dllimport) в объявлении функции. Этот модификатор оптимизирует импорт функции в приложении. По умолчанию шаблон нового проекта для библиотеки DLL добавляет символ *PROJECTNAME*\_EXPORTS в список определенных символов для проекта DLL. В этом примере символ MATHFUNCSDLL\_EXPORTS определяется при сборке проекта.

*Файл MyInfoHelperD.cpp*

namespace MyInfoHelperD

{

\_\_int64 convertToGB(\_\_int64 size)

{

return size / (1024 \* 1024 \* 1024);

}

int MyInfoHelperD::fileSize(wchar\_t\* path) {

HANDLE hFile = CreateFile(path,

GENERIC\_READ, // open for reading

0, // do not share

NULL, // default security

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL, // normal file

NULL);

DWORD size = GetFileSize(hFile, NULL);

CloseHandle(hFile);

if (size == INVALID\_FILE\_SIZE && (size = GetLastError()) != NO\_ERROR)

return -1;

else

return size;

}

void MyInfoHelperD::totalHDInfo() {

cout << "Total HD info: : " << endl;

int sz = GetLogicalDriveStrings(NULL, 0);

TCHAR\* szLogicalDrives = new TCHAR[sz];

GetLogicalDriveStrings(sz, szLogicalDrives);

while (\*szLogicalDrives)

{

TCHAR szDisk[80];

lstrcpy(szDisk, szLogicalDrives);

UINT uDriveType = GetDriveType(szDisk);

if (uDriveType == DRIVE\_FIXED)

{

wcout << szDisk << endl;

\_\_int64 free = 0, total = 0;

GetDiskFreeSpaceEx(szDisk, NULL, (PULARGE\_INTEGER)&total, (PULARGE\_INTEGER)&free);

cout << "Total : " << convertToGB(total) << " GB" << endl;

cout << "Free : " << convertToGB(free) << " GB" << endl;

cout << "Used : " << convertToGB(total - free) << " GB" << endl;

}

while (\*szLogicalDrives) szLogicalDrives++;

szLogicalDrives++;

}

}

}

Теперь можно скомпилировать динамическую библиотеку, в отдельный файл с расширением dll, который может использоваться другими программами.

Для того что бы использовать динамическую библиотеку в консольном приложении, необходимо в Visual Studio установить соответствующую связь между консольным проектом и динамической библиотекой. Так же при сборке консольного приложения необходимо указать заголовочный файл, указывающий на динамическую библиотеку.

Далее в главный файл консольного приложения подключить динамическую библиотеку и реализовать её функции. Код консольного приложения приведен ниже:

#include "stdafx.h"

#include "MyInfoHelperS.h"

#include "MyInfoHelperD.h"

#include <iostream>

using namespace std;

int main()

{

wchar\_t\* path = L"C:\\vm\\OS X Yosemite\\OS X Yosemite.vmdk";

int size = MyInfoHelperS::MyInfoHelperS::fileSize(path);

printf("MyInfoHelper static\n");

wcout << "File size of " << path << " - " << size << endl;

MyInfoHelperS::MyInfoHelperS::totalHDInfo();

size = MyInfoHelperD::MyInfoHelperD::fileSize(path);

printf("\nMyInfoHelper dynamic\n");

wcout << "File size of " << path << " - " << size << endl;

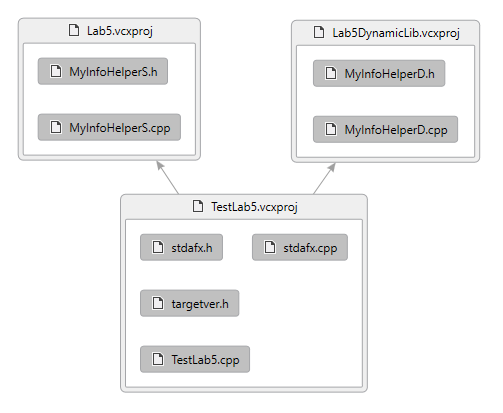
MyInfoHelperD::MyInfoHelperD::totalHDInfo();

system("pause");

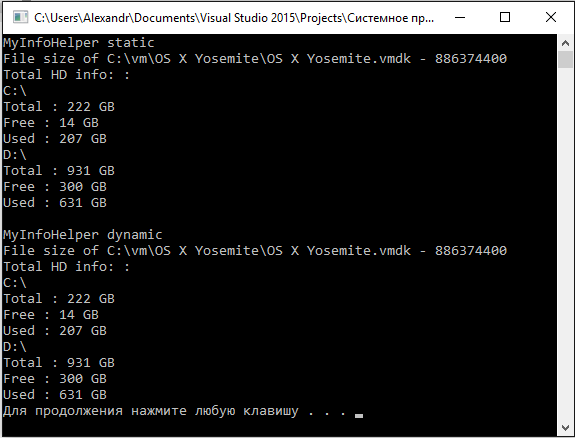
return 0;

}

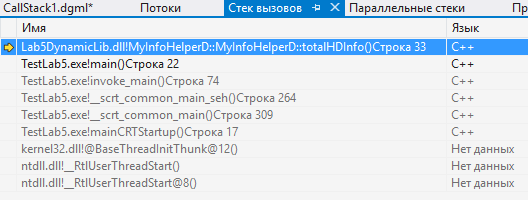
После выполнения этих действий можно собирать решение целиком и запускать консольное приложение. При этом будут вызываться функции как статической, так и динамической библиотек. Ниже показана связь между консольным приложением и соответствующих библиотек в общем решении.



После выполнения консольного приложения получены следующие результаты:

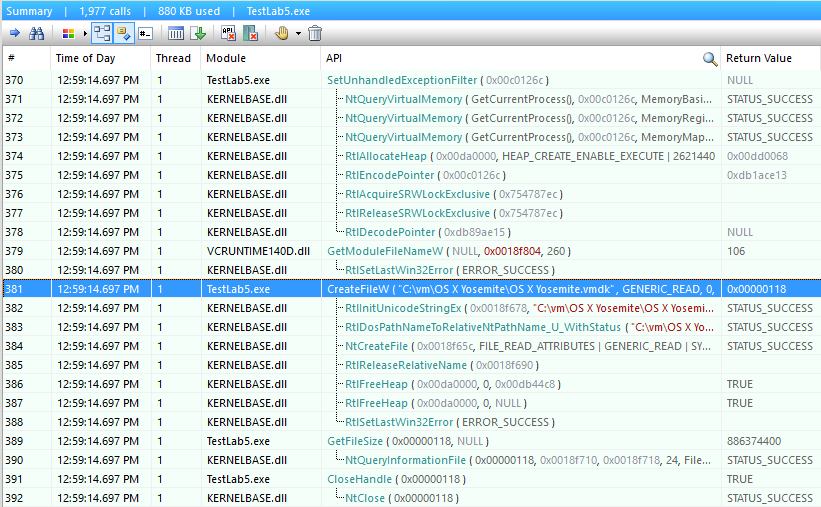


Ниже приведен стек вызовов, в котором видно, что функция totalHDInfo находится в файле Lab5DynamicLib.dll. Динамические библиотеки подключаются к исполняемому модулю во время его работы.

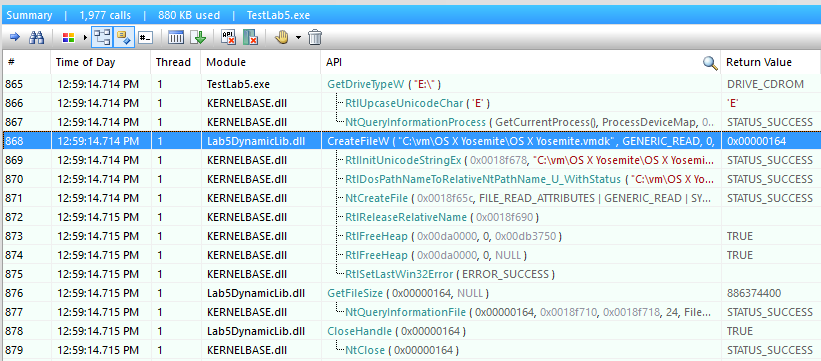


С помощью утилиты **API Monitor** можно выполнить профилирование консольного приложение, используещее статическую и динамическую библиотеки. Ниже приведен пример работы программы и список функций, вызванных из разработанной библиотеки.

В первую очередь в консольном приложении используются функции статической библиотеки. На пример (строка 381), WIN API функция CreateFileW(“…”) вызывается из модуля TestLab5.exe, т.е. из самой программы. Это означает что статическая библиотека скомпилирована вместе с консольным приложением.



Затем после используется динамическая библиотека. Вызов её функций происходит из файла самой библиотеки. На пример (строка 868), WIN API функция CreateFileW(“…”) вызывается из модуля Lab5DynamicLib.dll. Это означает, что консольное приложение динамически вызывает соответствующие функции у библиотеки, т.е. динамическая библиотека предстовляет собой отдельную программу, которую могут использовать дургие приложения. Динамические библиотеки не связаны с исполняемым файлом, поэтому все API функции вызываются непосредственно динамическими библиотеками.



# Создание статической библиотеки в Linux

Создадим статическую библиотеку для использования в Linux, используя функции, реализованные для Windows. Изменим исходный код программ в соответствии стандартам языка С.

Объявление библиотечных функций находится в заголовочном файле lab5static.h. В файле lab5static.m будет реализация функций: fileSize и printMemoryInfo. Ниже приведен код статической библиотеки:

*Файл lab5static.h*

int fileSize(char\* path);

void printMemoryInfo();

*Файл lab5static.c*

#include "sys/statfs.h"

#include "sys/stat.h"

#include "stdio.h"

#define GB (1024 \* 1024 \* 1024)

int fileSize(char\* path)

{

int result = 0;

struct stat st;

stat(path, &st);

result = st.st\_size;

return result;

}

void printMemoryInfo()

{

struct statfs info;

if (statfs("/", &info) == -1)

printf("error");

else {

double total = (double)(info.f\_bsize\*info.f\_blocks) / GB;

double available = (double)(info.f\_bsize\*info.f\_bfree) / GB;

double used = total - available;

double usedPercentage = (used / total) \* 100;

printf("Total : %.2f Gb; ", total);

printf("available : %.2f Gb; ", available);

printf("used : %.2f Gb (%.2f%%)\n", used, usedPercentage);

}

}

Напишем следующую программу, которая будет использовать функции из статической библиотеки. Ниже приведен код этой программы:

#include "lab5static.h"

#include "stdio.h"

int main()

{

char\* filePath = "/home/alex/Documents/Labs/Lab1/out.txt";

int size = fileSize(filePath);

printf("Hello Alex\n");

printf("%s fileSize: %d bytes\n", filePath, size);

printMemoryInfo();

return 0;

}

Далее произведем сборку статической библиотеки. Используя утилиту gcc создадим объектные файлы:

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 5/Linux$ **gcc -Wall -g -c -o lab5static.o lab5static.c**

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 5/Linux$ ls -la

drwxrwxr-x 2 alex alex 4096 Feb 14 15:22 .

drwxrwxr-x 5 alex alex 4096 Feb 14 14:12 ..

-rw-rw-r-- 1 alex alex 678 Feb 14 15:21 lab5static.c

-rw-rw-r-- 1 alex alex 49 Feb 14 15:20 lab5static.h

-rw-rw-r-- 1 alex alex 7480 Feb 14 15:22 lab5static.o

-rw-rw-r-- 1 alex alex 234 Feb 14 15:21 lab5testLib.c

Ключ –c означает, что необходима только компиляция. Из исходного файла **lab5static.c** программы создается объектный файл **lab5static.o.** Компоновка не производится.

Затем создадим статическую библиотеку lab5static.a:

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 5/Linux$ **ar rcs lab5static.a lab5static.o**

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 5/Linux$ ls -la

drwxrwxr-x 2 alex alex 4096 Feb 14 15:26 .

drwxrwxr-x 5 alex alex 4096 Feb 14 14:12 ..

-rw-rw-r-- 1 alex alex 7646 Feb 14 15:26 **lab5static.a**

-rw-rw-r-- 1 alex alex 678 Feb 14 15:21 lab5static.c

-rw-rw-r-- 1 alex alex 49 Feb 14 15:20 lab5static.h

-rw-rw-r-- 1 alex alex 7480 Feb 14 15:22 lab5static.o

-rw-rw-r-- 1 alex alex 234 Feb 14 15:21 lab5testLib.c

Получен архив lab5static.a статической библиотеки. Название библиотеки не является корректным, т.к. принято обозначать статические библиотеки приставкой lib. Далее скомпилируем вышеприведенную программы со статической библиотекой. И запустим исполняемый файл a.out.

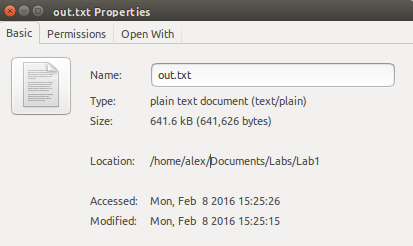
alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 5/Linux$ **gcc -static lab5testLib.c lab5static.a**

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 5/Linux$ ./a.out Hello Alex

/home/alex/Documents/Labs/Lab1/out.txt fileSize: 641626 bytes

Total : 18.58 Gb; available : 12.61 Gb; used : 5.97 Gb (32.13%)

Программа успешно скомпилировалась и получены ожидаемые результаты. Исполняемый файл a.out может использоваться без необходимости lab5static.a. Ниже приведен текстовый файл, размер которого вычисляется в программе.



Для сборки программы со статическими библиотеками требуется выполнения нескольких последовательных действ с компиляцией, что замедляет процесс компиляции конечной программы. Более удобный вариант будет с созданием Makefile в который записать последовательность сборки приложения и статических библиотек. Утилита make автоматически выполнит все промежуточные действия с компилятором и выдаст исполняемый файл. Ниже приведен пример реализации makefile:

binary: lab5static.a

gcc -static lab5testLib.c lab5static.a

lab5static.a: lab5static.o

ar cr lab5static.a lab5static.o

lab5static.o: lab5static.c

gcc -c lab5static.c

clean:

rm -f \*.o \*.a binary

# Создание динамических библиотек в Linux

Создадим разделяемую (динамическую) библиотеку (файл SO) для использования в Linux.

Динамические библиотеки полезны в случаях, если:  
- Важно не перекомпилировать всю программу, а только перекомпилировать ту часть, которая реализует определенные функции - тогда эти функции выносятся в динамическую библиотеку;   
- Кроме того, динамические библиотеки позволяют экономить место на жестком диске и в оперативной памяти, т.к. одну библиотеку могут использовать несколькими программами.

В Linux, обычно, динамические библиотеки имеют расширение ".so".

Исходный код динамический библиотеки аналогичен коду статической. Отличия заключаются только в процессе компиляции и линковки.

Т.к. для компиляции используется makefile, то необходимо обновить его инструкции для сборки динамических библиотек.

binary: lab5static.a

gcc -g lab5testLib.c lab5static.so

export LD\_LIBRARY\_PATH=/home/alex/Dropbox/политех/Системное\ программирование/Л.\ р.\ №\ 5/Linux:$LD\_LIBRARY\_PATH

lab5static.a: lab5static.o

gcc -shared -o lab5static.so lab5static.o

lab5static.o: lab5static.c

gcc –c -g -fPIC lab5static.c

clean:

rm -f \*.o \*.a binary

-fPIC - использовать относительную адресацию в переходах подпрограмм - во избежание конфликтов при динамическом связывании. Так как несколько различных программ могут использовать одну библиотеку, и каждая из них располагается в различном адресном пространстве. Поэтому требуется, чтобы переходы в функциях библиотеки (операции goto на ассемблере) использовали не абсолютную адресацию, а относительную. То есть генерируемый компилятором код должен быть независимым от адресов, такая технология получила название PIC - Position Independent Code.

-shared - предписывает создать динамическую (т.е. "разделяемую") библиотеку.

Необходимо установить значение для переменной окружения LD\_LIBRARY\_PATH, путь к директории в которой находится динамическая библиотека. Или при компиляции можно использовать флаги -Wl,-rpath для установки директорий линковщику.

После выполнения компиляции и запуска программы получены следующие результаты.

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 5/Linux$ make

gcc -c -fPIC lab5static.c

gcc -shared -o lab5static.so lab5static.o

gcc lab5testLib.c lab5static.so

export LD\_LIBRARY\_PATH=/home/alex/Dropbox/политех/Системное\ программирование/Л.\ р.\ №\ 5/Linux:D\_LIBRARY\_PATH

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 5/Linux$ ls -la

total 88

drwxrwxr-x 2 alex alex 4096 Feb 15 17:19 .

drwxrwxr-x 5 alex alex 4096 Feb 15 16:14 ..

-rwxrwxr-x 1 alex alex 8712 Feb 15 17:19 a.out

-rw-rw-r-- 1 alex alex 678 Feb 15 15:21 lab5static.c

-rw-rw-r-- 1 alex alex 49 Feb 15 15:20 lab5static.h

-rw-rw-r-- 1 alex alex 2696 Feb 15 17:19 lab5static.o

-rwxrwxr-x 1 alex alex 8320 Feb 15 17:19 lab5static.so

-rw-rw-r-- 1 alex alex 259 Feb 15 15:35 lab5testLib.c

-rw-rw-r-- 1 alex alex 376 Feb 15 17:15 makefile

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 5/Linux$ ./a.out

Hello Alex

/home/alex/Documents/Labs/Lab1/out.txt fileSize: 641626 bytes

Total : 18.58 Gb; available : 12.60 Gb; used : 5.97 Gb (32.16%)

Полученные результаты полностью идентичны с статической библиотекой.

С помощью утилиты ldd проверим какие динамические библиотеки, использует полученный исполняемый файл a.out:

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 5/Linux$ ldd ./a.out

linux-vdso.so.1 => (0x00007ffc937f0000)

**lab5static.so** => /home/alex/Dropbox/политех/Системное программирование/Л. р. № 5/Linux/lab5static.so (0x00007f524014a000)

libc.so.6 => /lib/x86\_64-linux-gnu/libc.so.6 (0x00007f523fd69000)

/lib64/ld-linux-x86-64.so.2 (0x000055993f6d6000)

Видно что в программе используется динамическая библиотека lib5static.so. Статические библиотеки утилитой ldd не выводятся. Ниже приведен дизассемблированный код, в котором показаны инструкции для переключения в динамическую библиотеку.

Dump of assembler code for function main:

5 {

0x0000000000400786 <+0>: push %rbp

0x0000000000400787 <+1>: mov %rsp,%rbp

0x000000000040078a <+4>: sub $0x10,%rsp

6 char\* filePath = "/home/alex/Documents/Labs/Lab1/out.txt";

0x000000000040078e <+8>: movq $0x400868,-0x8(%rbp)

7 int size = fileSize(filePath);

0x0000000000400796 <+16>: mov -0x8(%rbp),%rax

0x000000000040079a <+20>: mov %rax,%rdi

0x000000000040079d <+23>: **callq 0x400640 <fileSize@plt>**

0x00000000004007a2 <+28>: mov %eax,-0xc(%rbp)

8 printf("Hello Alex\n");

0x00000000004007a5 <+31>: mov $0x40088f,%edi

0x00000000004007aa <+36>: callq 0x400630 <puts@plt>

Далее приведем пример работы программы с помощью утилиты strace. Strace возвращает список системных вызовов.

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 5/Linux$ strace ./a.out

execve("./a.out", ["./a.out"], [/\* 65 vars \*/]) = 0

brk(0) = 0x14e1000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5a88dd9000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

**open("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\270\321\200\320\276\320\262\320\260\320\275\320\270\320\265/\320\233. \321\200. \342\204\226 5/Linux/tls/x86\_64/lab5static.so", O\_RDONLY|O\_CLOEXEC) = -1** ENOENT (No such file or directory)

stat("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\270\321\200\320\276\320\262\320\260\320\275\320\270\320\265/\320\233. \321\200. \342\204\226 5/Linux/tls/x86\_64", 0x7ffede35d340) = -1 ENOENT (No such file or directory)

open("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\270\321\200\320\276\320\262\320\260\320\275\320\270\320\265/\320\233. \321\200. \342\204\226 5/Linux/tls/lab5static.so", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\270\321\200\320\276\320\262\320\260\320\275\320\270\320\265/\320\233. \321\200. \342\204\226 5/Linux/tls", 0x7ffede35d340) = -1 ENOENT (No such file or directory)

open("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\270\321\200\320\276\320\262\320\260\320\275\320\270\320\265/\320\233. \321\200. \342\204\226 5/Linux/x86\_64/lab5static.so", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\270\321\200\320\276\320\262\320\260\320\275\320\270\320\265/\320\233. \321\200. \342\204\226 5/Linux/x86\_64", 0x7ffede35d340) = -1 ENOENT (No such file or directory)

open("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\270\321\200\320\276\320\262\320\260\320\275\320\270\320\265/\320\233. \321\200. \342\204\226 5/Linux/lab5static.so", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\7\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0775, st\_size=8320, ...}) = 0

mmap(NULL, 2101336, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5a889b6000

mprotect(0x7f5a889b7000, 2093056, PROT\_NONE) = 0

mmap(0x7f5a88bb6000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0) = 0x7f5a88bb6000

close(3) = 0

open("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\270\321\200\320\276\320\262\320\260\320\275\320\270\320\265/\320\233. \321\200. \342\204\226 5/Linux/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

open("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265/tls/x86\_64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265/tls/x86\_64", 0x7ffede35d310) = -1 ENOENT (No such file or directory)

open("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265/tls/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265/tls", 0x7ffede35d310) = -1 ENOENT (No such file or directory)

open("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265/x86\_64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265/x86\_64", 0x7ffede35d310) = -1 ENOENT (No such file or directory)

open("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/home/alex/Dropbox/\320\277\320\276\320\273\320\270\321\202\320\265\321\205/\320\241\320\270\321\201\321\202\320\265\320\274\320\275\320\276\320\265", 0x7ffede35d310) = -1 ENOENT (No such file or directory)

open("/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=86295, ...}) = 0

mmap(NULL, 86295, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f5a88dc3000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

open("/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`\v\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=1869392, ...}) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5a88dc2000

mmap(NULL, 3972864, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5a885ec000

mprotect(0x7f5a887ac000, 2097152, PROT\_NONE) = 0

mmap(0x7f5a889ac000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c0000) = 0x7f5a889ac000

mmap(0x7f5a889b2000, 16128, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5a889b2000

close(3) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5a88dc1000

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5a88dc0000

arch\_prctl(ARCH\_SET\_FS, 0x7f5a88dc1700) = 0

mprotect(0x7f5a889ac000, 16384, PROT\_READ) = 0

mprotect(0x7f5a88bb6000, 4096, PROT\_READ) = 0

mprotect(0x600000, 4096, PROT\_READ) = 0

mprotect(0x7f5a88ddb000, 4096, PROT\_READ) = 0

munmap(0x7f5a88dc3000, 86295) = 0

stat("/home/alex/Documents/Labs/Lab1/out.txt", {st\_mode=S\_IFREG|0664, st\_size=641626, ...}) = 0

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 19), ...}) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5a88dd8000

write(1, "Hello Alex\n", 11Hello Alex

) = 11

write(1, "/home/alex/Documents/Labs/Lab1/o"..., 62/home/alex/Documents/Labs/Lab1/out.txt fileSize: 641626 bytes

) = 62

statfs("/", {f\_type="EXT2\_SUPER\_MAGIC", f\_bsize=4096, f\_blocks=4869551, f\_bfree=3303646, f\_bavail=3050526, f\_files=1245184, f\_ffree=1026870, f\_fsid={1631239308, -1817461388}, f\_namelen=255, f\_frsize=4096}) = 0

write(1, "Total : 18.58 Gb; available : 12"..., 64Total : 18.58 Gb; available : 12.60 Gb; used : 5.97 Gb (32.16%)

) = 64

exit\_group(0) = ?

+++ exited with 0 +++

В процессе работы программы используется динамическая загрузка библиотеки: open("/home/alex/Dropbox/…/Linux/tls/x86\_64/lab5static.so", O\_RDONLY|O\_CLOEXEC) = -1. Значит, библиотека подгружается в ходе работы исполняемого файла.