Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе №5
Курс: Системное программирование
Тема: Создание статических и динамических библиотек

Выполнил студент группы 13541/3 _____ Д.В. Круминьш
(подпись)

Преподаватель _____ Е.В. Душутина
(подпись)

Санкт-Петербург
2017 г.

Содержание

Постановка задачи

В данной работе будут рассмотрены статические и динамические библиотеки в операционной системе windows и linux.

Подготовка к выполнению работы

2.1 Статические и динамические библиотеки

Использование библиотек является хорошим способом повторного использования кода. Вместо того чтобы каждый раз реализовывать одни и те же подпрограммы для обеспечения той или иной функциональности в каждом создаваемом приложении, их можно создать единожды и затем вызывать из приложений. Различают **статические** и **динамические** библиотеки. Основное отличие состоит в способе их использования. Код, подключенный из статической библиотеки, становится частью приложения — для использования кода не нужно устанавливать еще какой-либо файл, а для динамических библиотек это внешний файл(например формата dll для windows), который подключается в случае необходимости.

2.2 Сведения о системе Windows

Работа производилась на реальной системы, с параметрами представленными ниже.

Элемент	**Значение**
Имя ОС	Майкрософт Windows 10 Pro (Registered Trademark)
Версия	10.0.14393 Сборка 14393
Дополнительное описание ОС	Недоступно
Изготовитель ОС	Microsoft Corporation
Имя системы	USER-PC
Изготовитель	HP
Модель	OMEN by HP Laptop 15-ce0xx
Тип	Компьютер на базе x64
SKU системы	1ZB00EA#ACB
Процессор	Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz, 2496 МГц, ядер: 4, логических процессоров: 4
Версия BIOS	American Megatrends Inc. F.04, 10.05.2017
Версия SMBIOS	3.0
Версия встроенного контроллера	40.20
Режим BIOS	Устаревший
Изготовитель основной платы	HP

Модель основной платы	Недоступно
Имя основной платы	Основная плата
Роль платформы	Мобильный
Состояние безопасной загрузки	Не поддерживается
Конфигурация PCR7	Привязка невозможна
Папка Windows	C:\Windows
Системная папка	C:\Windows\system32
Устройство загрузки	\Device\HarddiskVolume1
Язык системы	Россия
Аппаратно-зависимый уровень (HAL)	Версия = ”10.0.14393.1378”
Имя пользователя	USER-PC\Tom
Часовой пояс	RTZ 2 (зима)
Установленная оперативная память (RAM)	8,00 ГБ
Полный объем физической памяти	7,87 ГБ
Доступно физической памяти	3,54 ГБ
Всего виртуальной памяти	12,6 ГБ
Доступно виртуальной памяти	6,82 ГБ
Размер файла подкачки	4,75 ГБ
Файл подкачки	C:\pagefile.sys

Таблица 2.1: Информация об используемой системе Windows

Для разработки использовалась Microsoft Visual Studio Enterprise 2017 (Версия 15.3.0).

2.3 Сведения о системе Linux

Все опыты проводятся на виртуальной системе(64 битной), сведения о которой приведены в листинге 3.1.

```
root@kali:~/Desktop/testFolder# cat /etc/*release*
DISTRIB_ID=Kali
DISTRIB_RELEASE=kali-rolling
DISTRIB_CODENAME=kali-rolling
DISTRIB_DESCRIPTION="Kali GNU/Linux Rolling"
PRETTY_NAME="Kali GNU/Linux Rolling"
NAME="Kali GNU/Linux"
ID=kali
VERSION="2017.2"
```

```
10 VERSION_ID="2017.2"
11 ID_LIKE=debian
12 ANSI_COLOR="1;31"
13 HOME_URL="http://www.kali.org/"
14 SUPPORT_URL="http://forums.kali.org/"
15 BUG_REPORT_URL="http://bugs.kali.org/"
16
17 root@kali:~# uname -r
18 4.12.0-kali1-amd64
19
20 root@kali:~# lscpu
21 Architecture:          x86_64
22 CPU op-mode(s):        32-bit, 64-bit
23 Byte Order:            Little Endian
24 CPU(s):                4
25 On-line CPU(s) list:   0-3
26 Thread(s) per core:    1
27 Core(s) per socket:    2
28 Socket(s):             2
29 NUMA node(s):          1
30 Vendor ID:             GenuineIntel
31 CPU family:            6
32 Model name:            Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz
33 Stepping:              9
34 CPU MHz:               2495.998
35 Hypervisor vendor:     VMware
36 Virtualization type:   full
37 L1d cache:             32K
38 L1i cache:             32K
39 L2 cache:              256K
40 L3 cache:              6144K
41 NUMA node0 CPU(s):     0-3
```

Листинг 2.1: Информация об используемой системе Linux

Работа происходила с использованием:

- Текстового редактора Sublime text(версии 3.0, сборка 3143);
- Компилятора gcc(версия Debian 7.2.0-4);
- Отладчика GNU Debugger(версия Debian 7.12-6).

Windows

3.1 Статическая библиотека

Создадим статическую библиотеку, которая реализует функции работы с файлами, а именно:

- Функция вывода содержимого какого-либо каталога;
- Функция вывода даты и времени создания файла;
- Функция вывода размера файла;
- Функция вывода имен запущенных процессов и их идентификаторов.

3.1.1 Создание библиотеки

После создания консольного приложения в visual studio, необходимо в свойствах решения изменить тип конфигурации на **Статическая библиотека(.lib).**

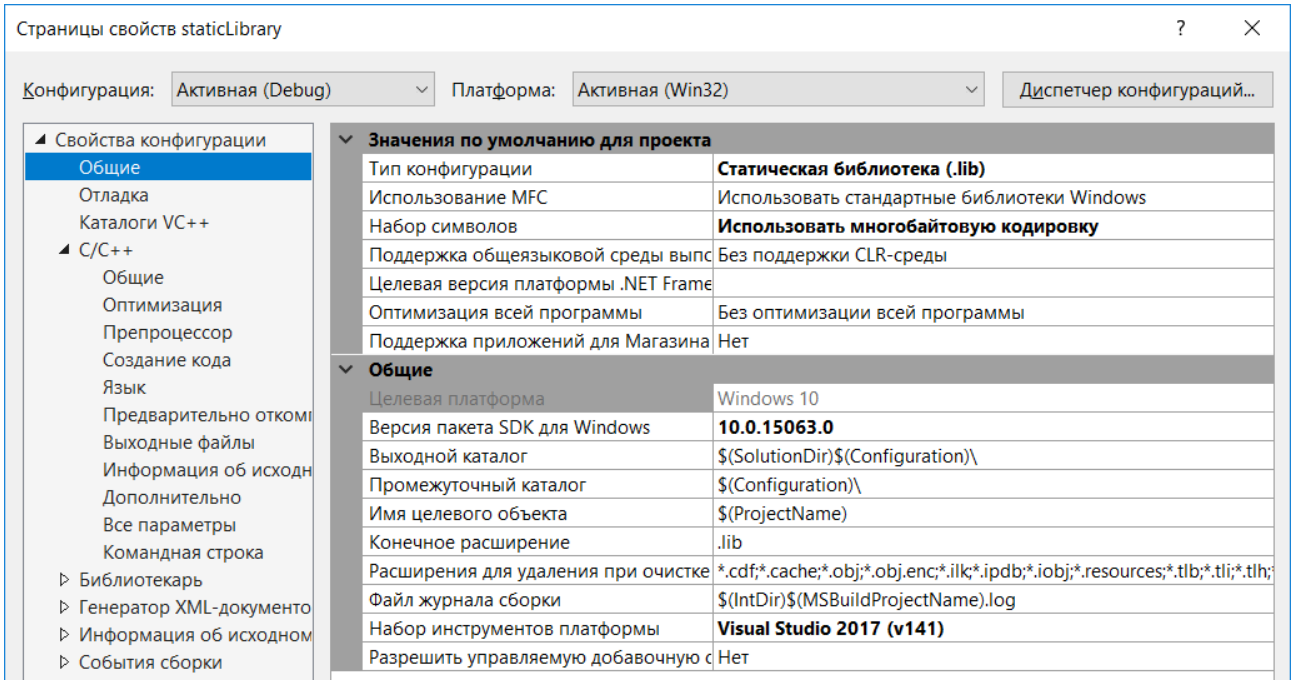

Рис. 3.1: Параметры проекта

Далее необходимо создать заголовочный файл и файл исходного кода, листинги которых представлен далее.

```
#include <windows.h>
namespace MyLib
{
    class MyLibFunctions
    {
    public:
        static void ls(LPCSTR path);
        static void dateFile(LPCSTR path);
        static void fileSize(LPCSTR path);
        static void processes();
    };
}
```

Листинг 3.1: MyLib.h

```
#define _CRT_SECURE_NO_WARNINGS
#include "MyLib.h"
#include <fstream>
#include <stdexcept>
#include <iostream>
#include <io.h>
#include <tlhelp32.h>
using namespace std;

namespace MyLib
{
    void MyLibFunctions::ls(LPCSTR str)
    {
        WIN32_FIND_DATA FindFileData;
        HANDLE hf;
        // в цикле выводи название файлов
        hf = FindFirstFile(str, &FindFileData);
        if (hf != INVALID_HANDLE_VALUE)
        {
            do
            {
                cout << FindFileData.cFileName << endl;
            } while (FindNextFile(hf, &FindFileData) != 0);
            FindClose(hf);
        }
    }

    void MyLibFunctions::dateFile(LPCSTR path)
    {
        HANDLE fH;
        FILETIME creationTime;
        SYSTEMTIME sysTime;
        // получаем дескриптор файла
        fH = CreateFile(path, GENERIC_READ, 0, 0, OPEN_EXISTING,
    ↪ FILE_ATTRIBUTE_NORMAL, 0);
        if (fH != INVALID_HANDLE_VALUE)
        {
            // получение времени создания файла
            GetFileTime(fH, &creationTime, 0, 0);
            FileTimeToSystemTime(&creationTime, &sysTime);
            cout << path << " " << sysTime.wDay << "." << sysTime.wMonth << "."
    ↪  << sysTime.wYear <<
                 " " << sysTime.wHour << ":" << sysTime.wHour << std::endl;
            CloseHandle(fH);
        }
```

```
44      }
45
46      void MyLibFunctions::fileSize(LPCSTR path)
47      {
48          HANDLE hFile;
49          DWORD lcFileSize;
50          // создаем дескриптор файла
51          hFile = CreateFile(path, GENERIC_READ, 0, 0, OPEN_EXISTING,
52              FILE_ATTRIBUTE_NORMAL, 0);
53          // получаем размер файла
54          lcFileSize = GetFileSize(hFile, 0);
55          CloseHandle(hFile);
56          cout << "File size: " << lcFileSize << " bytes"<< endl;
57
58      }
59
60      void MyLibFunctions::processes()
61      {
62          PROCESSENTRY32 pt;
63          // получение снапшота всех процессов
64          HANDLE hsnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
65          pt.dwSize = sizeof(PROCESSENTRY32);
66          if (Process32First(hsnap, &pt)) { // поиск необходимого процесса
67              do {
68                  printf("Proc name: %s, proc id: %ld\n", pt.szExeFile, pt.
    ↪ th32ProcessID);
69              } while (Process32Next(hsnap, &pt));
70          }
71          CloseHandle(hsnap);
72      }
73  }
```

Листинг 3.2: MyLib.cpp

При сборке библиотеки создается соответствующий файл с расширением **lib**.

```
1  1>------ Сборка начата: проект: staticLibrary, Конфигурация: Debug Win32 ------
2  1>MyLib.cpp
3  1>staticLibrary.vcxproj -> E:\study\s09\SystemProgramming\labs\libraries\
       ↪ project\p1\staticLibrary\Debug\staticLibrary.lib
4  ========== Сборка: успешно: 1, с ошибками: 0, без изменений: 0, пропущено: 0
       ↪ ==========
```

Листинг 3.3: Сборка библиотеки

3.1.2 Создание приложения

При создании приложения, которое будет использовать функциональность библиотеки, при создании необходимо в пункте **Решение**, выбрать пункт **Добавить в решение**.

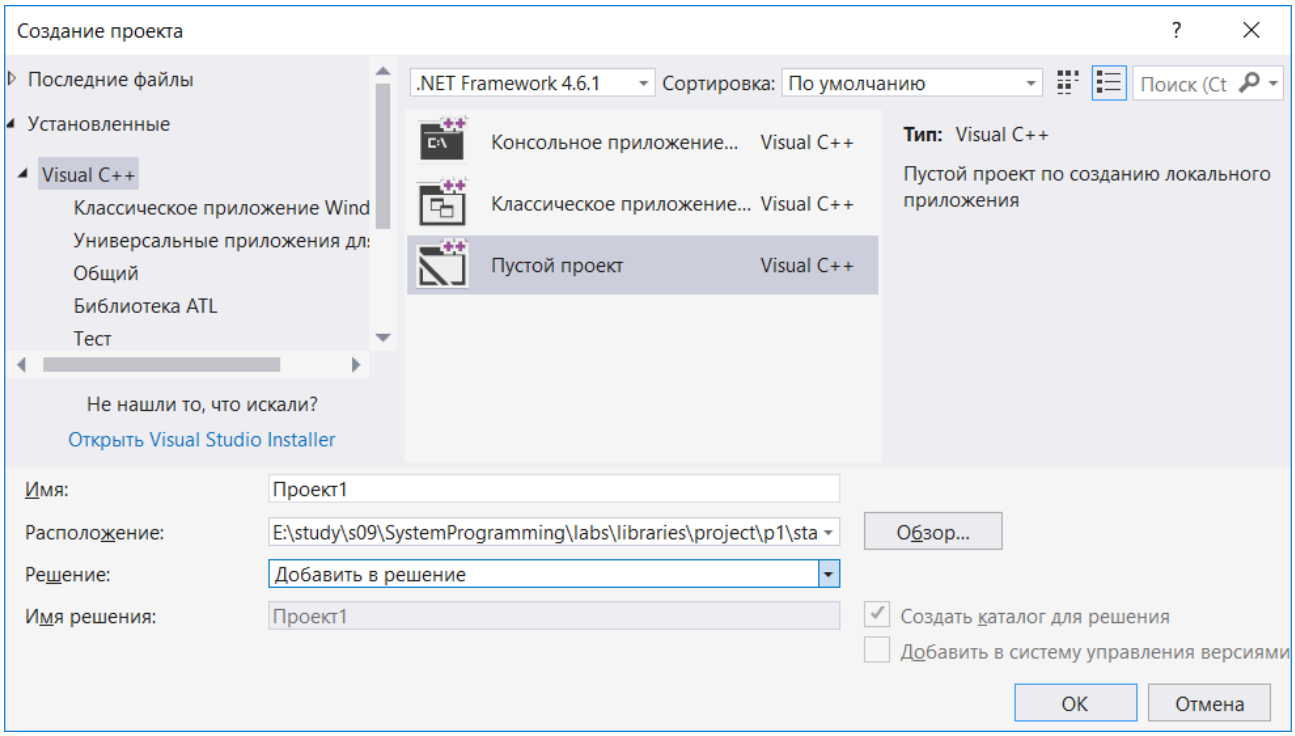

Рис. 3.2: Тип решения

Далее создать файл с исходным кодом, приведенном в листинге 3.4.

```
1  #include <iostream>
2
3  #include "MyLib.h"
4
5  using namespace std;
6
7  int main()
8  {
9      MyLib::MyLibFunctions().ls("E:\\*");
10     cout << endl;
11     MyLib::MyLibFunctions().dateFile("E:\\myServiceLog.txt");
12     cout << endl;
13     MyLib::MyLibFunctions().fileSize("E:\\myServiceLog.txt");
14     cout << endl;
15     MyLib::MyLibFunctions().processes();
16     return 0;
17 }
```

Листинг 3.4: MyProgram.cpp

Далее в созданном проекте, необходимо добавить ссылку, как показано на рисунке 3.3.

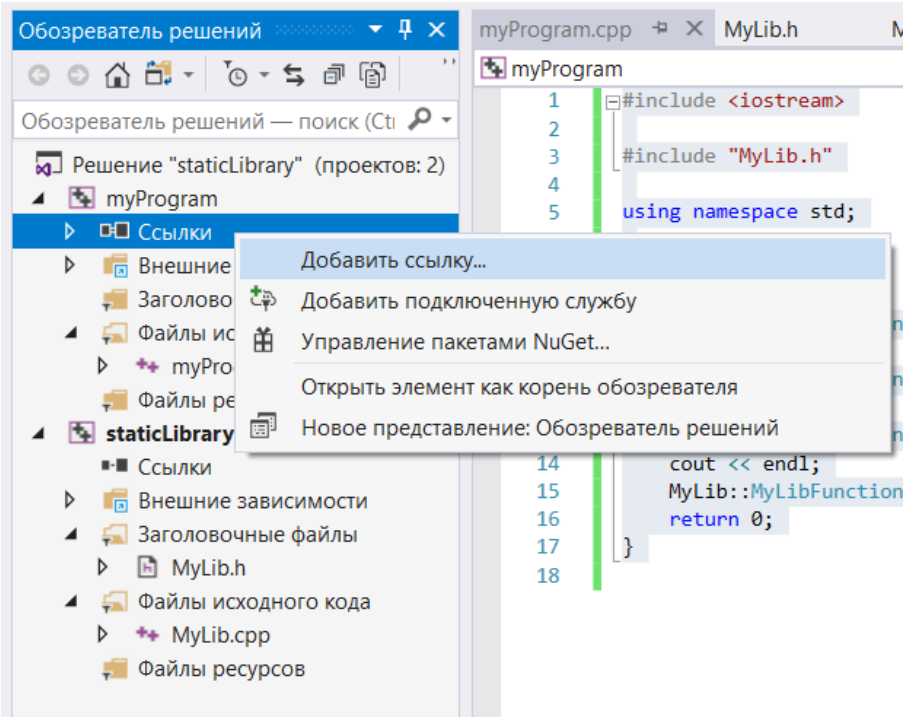

Рис. 3.3: Добавление ссылки

В появившемся окне выбрать библиотеку.

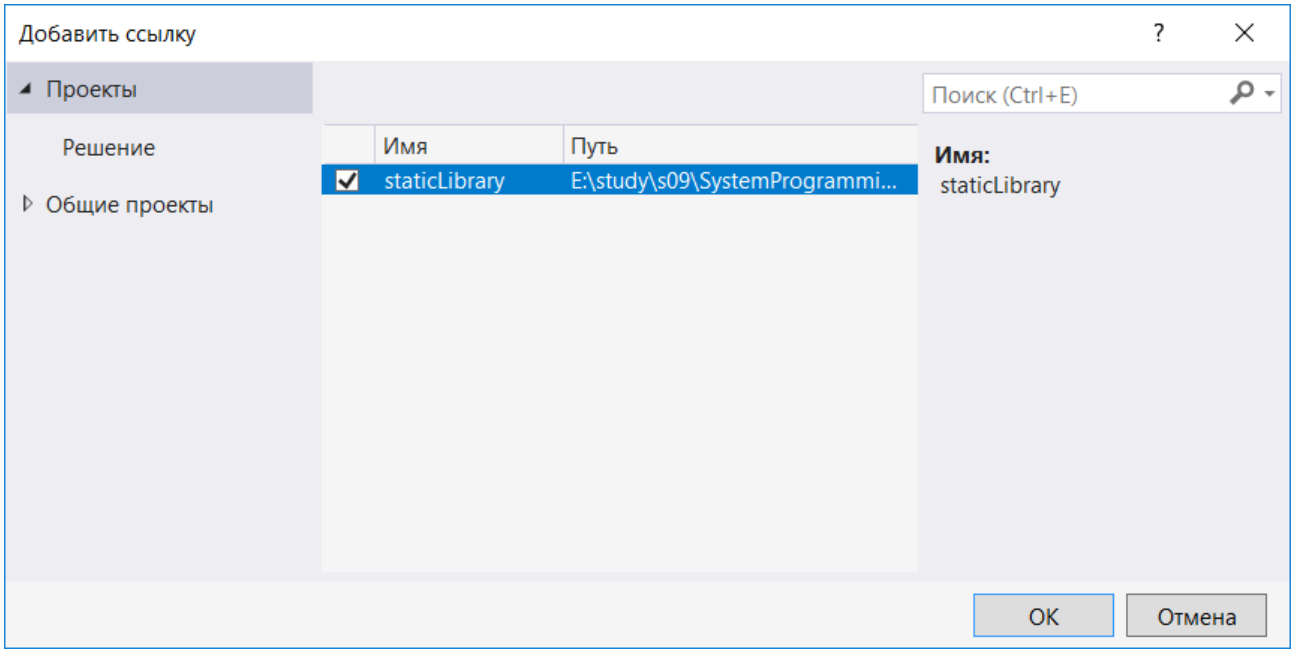

Рис. 3.4: Выбор ссылки

Конечная структура проекта должна выглядеть как на рисунке 3.5.

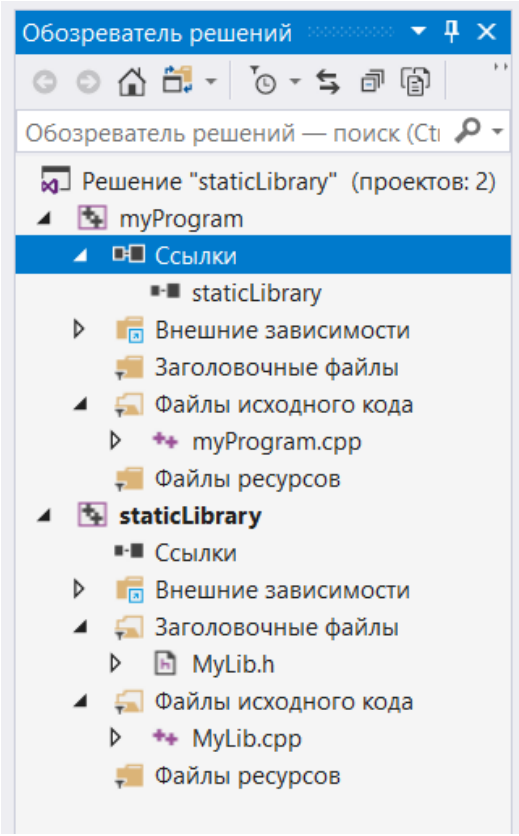

Рис. 3.5: Структура проекта

Теперь необходимо перейти в свойства решения, во вкладку **C/C++**, пункт **Общие**. В нем, в пункте **Дополнительные каталоги включаемых файлов** необходимо указать путь к проекту статической библиотеки.

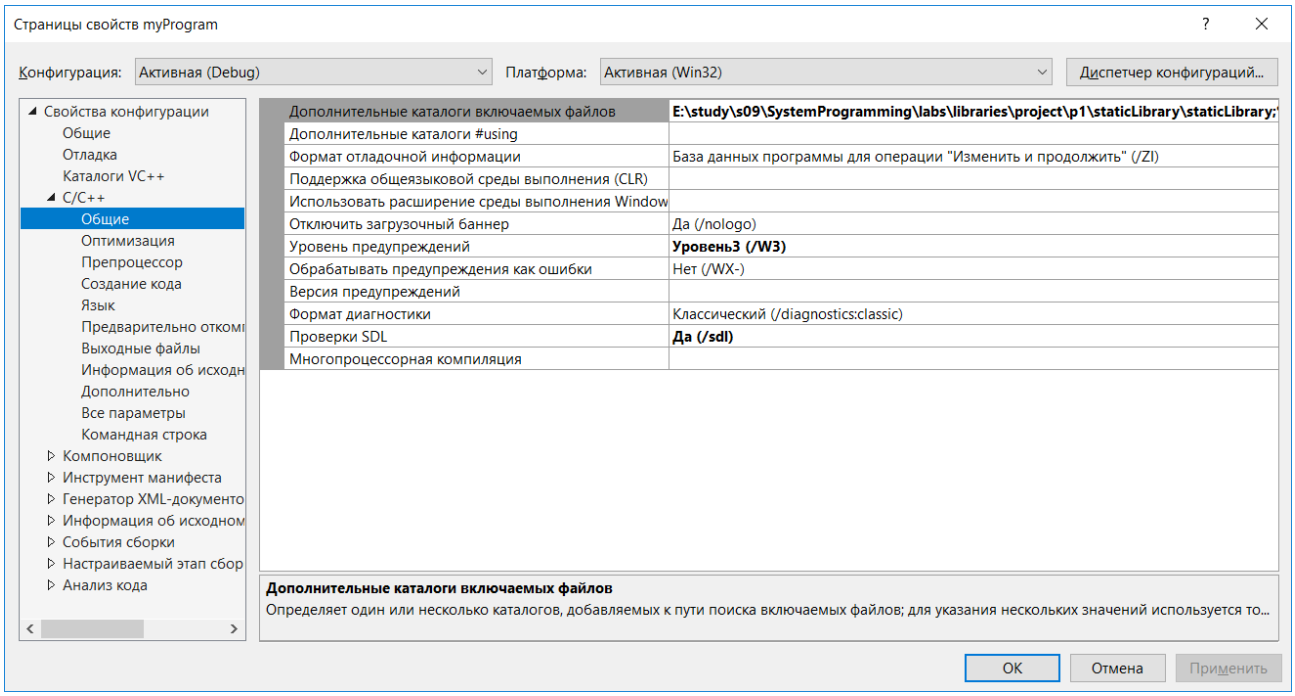

Рис. 3.6: Установка зависимости

Наконец, после этого можно собрать приложение.

3.1.3 Пример использования

Переходим в папку проекта с собранным приложением и запускаем его.

```
E:\study\s09\SystemProgramming\labs\libraries\project\p1\staticLibrary\Debug>
    ↪ myProgram.exe
$RECYCLE.BIN
bachelor_thesis
expirements
Games
log.txt
msdownld.tmp
myServiceLog.txt
Soft
study
System Volume Information
testFolder
testFolder2
torrent
work

E:\myServiceLog.txt 23.11.2017 15:15

File size: 25 bytes

Proc name: [System Process], proc id: 0
Proc name: System, proc id: 4
Proc name: smss.exe, proc id: 480
Proc name: csrss.exe, proc id: 700
Proc name: csrss.exe, proc id: 796
Proc name: wininit.exe, proc id: 820
Proc name: winlogon.exe, proc id: 880
Proc name: services.exe, proc id: 932
Proc name: lsass.exe, proc id: 948
Proc name: svchost.exe, proc id: 372
Proc name: svchost.exe, proc id: 532
Proc name: svchost.exe, proc id: 928
Proc name: svchost.exe, proc id: 1044
Proc name: dwm.exe, proc id: 1100
Proc name: svchost.exe, proc id: 1132
Proc name: svchost.exe, proc id: 1228
Proc name: WUDFHost.exe, proc id: 1284
Proc name: svchost.exe, proc id: 1340
Proc name: NVDisplay.Container.exe, proc id: 1412
Proc name: igfxCUIService.exe, proc id: 1460
Proc name: svchost.exe, proc id: 1556
...
```

Листинг 3.5: Выполнение программы

Как и ожидалось, все 4 функции успешно выполнились. Стоит отметить что если изменять и пересобрать библиотеку, то результат не изменится, так как библиотека статическая, а приложение не было пересобрано, что является основным доказательство что приложением используется статическая библиотека.

3.2 Динамическая библиотека

Поместив код в библиотеку DLL, экономится место в каждом приложении, которое ссылается на этот код, а так же сможете обновлять DLL без перекомпиляции всех приложений, чего не удалось сделать со статической библиотекой.

3.2.1 Создание библиотеки

Необходимо дополнить файл заголовка, в него дополняются функции **dllexport** и **dllimport**[1] для экспорта и импорта функций соответственно.

```
1  #include <windows.h>
2
3  #ifdef MYFUNCSDLL_EXPORTS
4  #define MYFUNCSDLL_API __declspec(dllexport)
5  #else
6  #define MYFUNCSDLL_API __declspec(dllimport)
7  #endif
8
9  namespace MyLib
10 {
11     class MyLibFunctions
12     {
13     public:
14         static MYFUNCSDLL_API void ls(LPCSTR path);
15         static MYFUNCSDLL_API void dateFile(LPCSTR path);
16         static MYFUNCSDLL_API void fileSize(LPCSTR path);
17         static MYFUNCSDLL_API void processes();
18     };
19 }
```

Листинг 3.6: Сборка библиотеки

Исходный код идентичен коду из листинга 3.2. Необходимо лишь изменить тип проекта на **Динамическая библиотека (.dll)**.

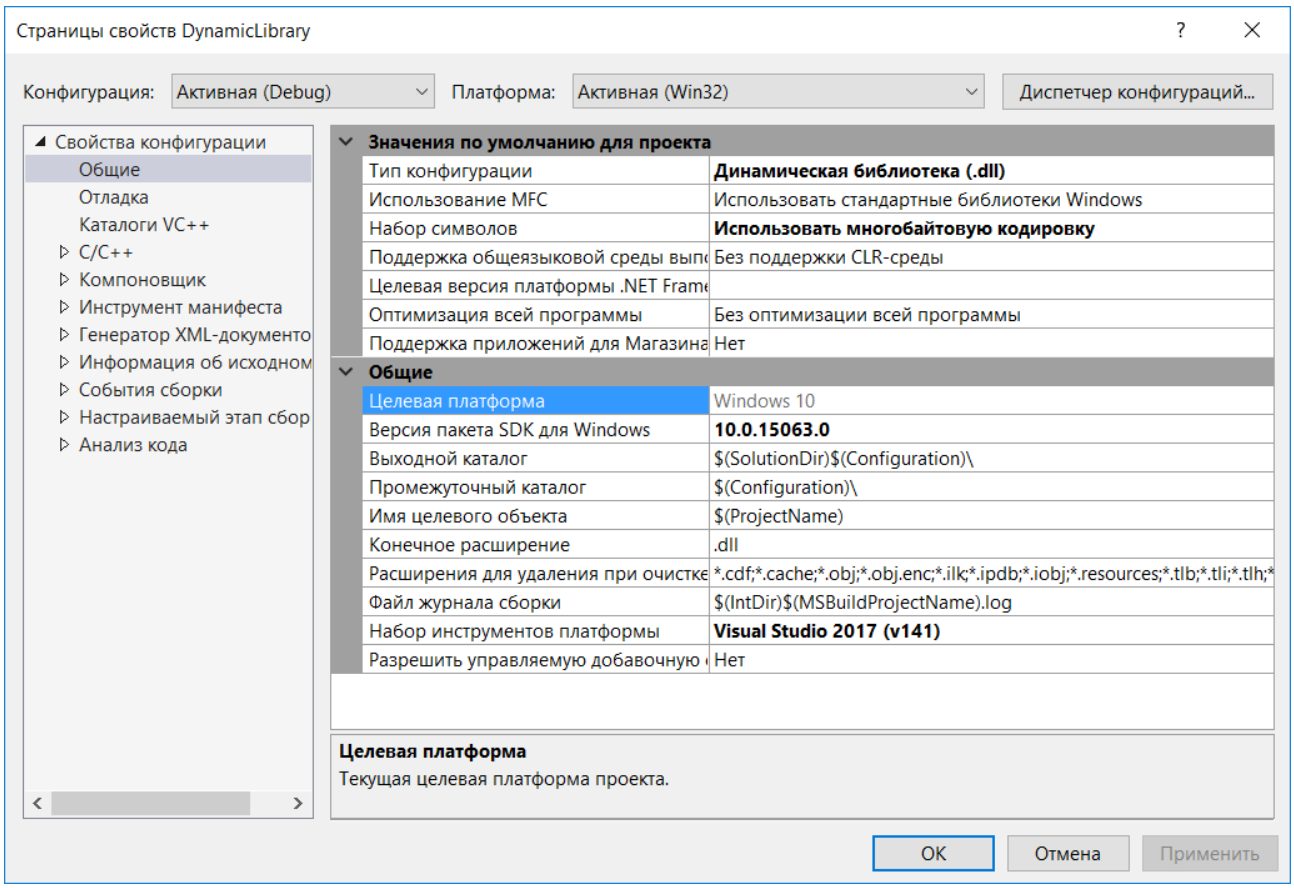

Рис. 3.7: Параметры проекта

Сборка проходит успешно и создает файл с расширением **dll**.

```
1>------ Сборка начата: проект: DynamicLibrary, Конфигурация: Debug Win32 ------
1>MyLib.cpp
1>DynamicLibrary.vcxproj -> E:\study\s09\SystemProgramming\labs\libraries\
    ↪ project\p2\DynamicLibrary\Debug\DynamicLibrary.dll
1>DynamicLibrary.vcxproj -> E:\study\s09\SystemProgramming\labs\libraries\
    ↪ project\p2\DynamicLibrary\Debug\DynamicLibrary.pdb (Partial PDB)
========== Сборка: успешно: 1, с ошибками: 0, без изменений: 0, пропущено: 0
    ↪ ==========
```

Листинг 3.7: Сборка библиотеки

3.2.2 Создание приложения

Нужно поставить тип решения на **Добавить в решение**.

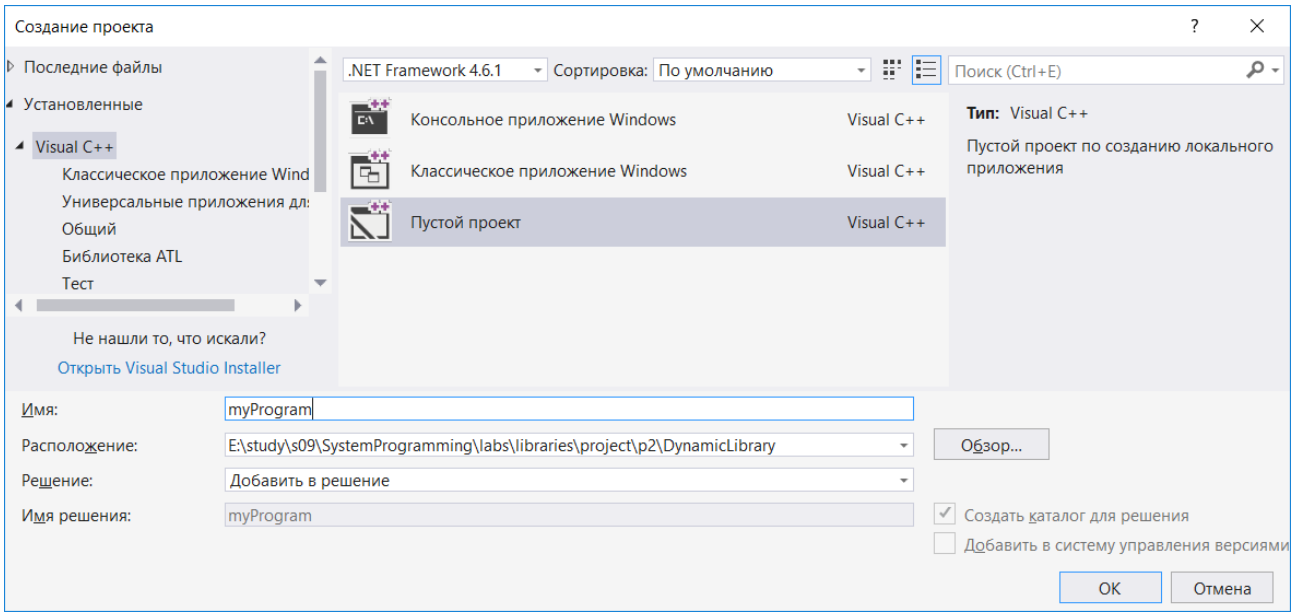

Рис. 3.8: Тип решения

Создать файл с исходным кодом из листинга 3.4. Также аналогично действиям для статической библиотеки, добавить ссылку и указать путь в пункте **Дополнительные каталоги включаемых файлов**.

Конечная структура должна выглядеть следующим образом:

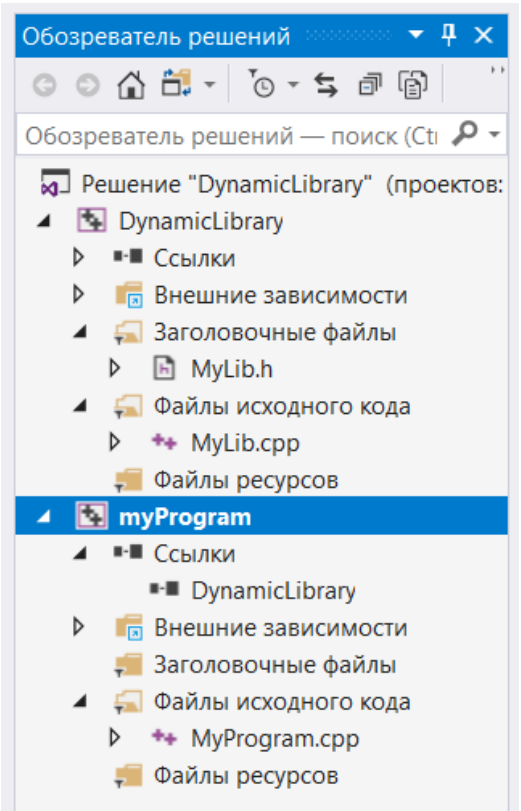

Рис. 3.9: Структура проекта

3.2.3 Пример использования

Результат выполнения не отличается.

```
1  E:\study\s09\SystemProgramming\labs\libraries\project\p1\staticLibrary\Debug>
       ↪ myProgram.exe
2  $RECYCLE.BIN
3  bachelor_thesis
4  expirements
5  Games
6  log.txt
7  msdownld.tmp
8  myServiceLog.txt
9  Soft
10 study
11 System Volume Information
12 testFolder
13 testFolder2
14 torrent
15 work
16
17 E:\myServiceLog.txt 23.11.2017 15:15
18
19 File size: 25 bytes
20
21 Proc name: [System Process], proc id: 0
22 Proc name: System, proc id: 4
23 Proc name: smss.exe, proc id: 480
24 Proc name: csrss.exe, proc id: 700
25 Proc name: csrss.exe, proc id: 796
26 Proc name: wininit.exe, proc id: 820
27 Proc name: winlogon.exe, proc id: 880
28 Proc name: services.exe, proc id: 932
29 Proc name: lsass.exe, proc id: 948
30 Proc name: svchost.exe, proc id: 372
31 Proc name: svchost.exe, proc id: 532
32 Proc name: svchost.exe, proc id: 928
33 Proc name: svchost.exe, proc id: 1044
34 Proc name: dwm.exe, proc id: 1100
35 Proc name: svchost.exe, proc id: 1132
36 Proc name: svchost.exe, proc id: 1228
37 Proc name: WUDFHost.exe, proc id: 1284
38 Proc name: svchost.exe, proc id: 1340
39 Proc name: NVDisplay.Container.exe, proc id: 1412
40 Proc name: igfxCUIService.exe, proc id: 1460
41 Proc name: svchost.exe, proc id: 1556
42 ...
```

Листинг 3.8: Выполнение программы

Дополнительно было произведено сравнение размеров конечных программ.
Статическая реализация - 49 152 байта.
Динамическая реализация - 38 400 байт.

Как и ожидалось, программа со статическим использование библиотеки, имеет больший размер.

Linux

Библиотека- это набор скомпонованных особым образом объектных файлов. Библиотеки подключаются к основной программе во время линковки.

Статическая библиотека - это архив объектных файлов, который подключается к программе во время линковки. Эффект такой же, как если бы подключали каждый из файлов отдельно.

Далее в разделе исходного кода, представлен исходный тексты программы, разделенной на несколько частей:

- **MyLib.h** - заголовок библиотеки;
- **main.c** - код основной программы, который вызвает функции из файлов **fileWrite.c** и **fileRead.c**;
- **fileWrite.c** - записывает в файл **tempFile.txt** некоторые символы;
- **fileRead.c** - читает файл **tempFile.txt** и выводит его содержимое в консоль.

4.1 Исходный код

```
1 //myLIb.h
2 void fileRead(void);
3 void fileWrite(void);
```

Листинг 4.1: MyLib.h

```
1 #include "myLib.h"
2
3 int main (void)
4 {
5         fileRead();
6         fileWrite();
7 }
```

Листинг 4.2: main.c

```
1  //fileWrite.c
2  #include <stdio.h>
3  #include "myLib.h"
4  void fileWrite(void)
5  {
6      FILE *fp = fopen("tempFile.txt", "wb");
7      for(int i=0;i<9;i++){
8          fputs("123456789\n"+i, fp);
9      }
10
```

```
11      fclose(fp);
12  }
```

Листинг 4.3: fileWrite.c

```
1   //fileRead.c
2   #include <stdio.h>
3   #include "myLib.h"
4
5   void fileRead () {
6       FILE *file;
7       char temp[1000];
8       file = fopen("tempFile.txt", "r");
9
10      while (fscanf (file, "%s", temp) != EOF) {
11          printf("%s\n", temp);
12      }
13  }
```

Листинг 4.4: fileRead.c

4.2 Динамическая библиотека

Для оптимизации работы был создан **makefile** следующего содержимого:

```
1   # Makefile for myDynLib project
2
3   binary: main.o libMyLib.so
4       gcc -o binary main.o -L. -lMyLib -Wl,-rpath,.
5
6   main.o: main.c
7       gcc -c main.c
8
9   libMyLib.so: fileRead.o fileWrite.o
10      gcc -shared -o libMyLib.so fileRead.o fileWrite.o
11
12  fileRead.o: fileRead.c
13      gcc -c -fPIC fileRead.c
14
15  fileWrite.o: fileWrite.c
16      gcc -c -fPIC fileWrite.c
17
18  clean:
19      rm -f *.o *.so binary
```

Листинг 4.5: makefile

Опция -l, переданная компилятору, обрабатывается и посылается линковщику для того, чтобы тот подключил к бинарнику библиотеку. Опция -L указывает линковщику, где ему искать библиотеку. Опциия –fPIC сообщает компилятору, что объектные файлы, полученные в результате компиляции должны содержать позиционно-независимый код (PIC - Position Independent Code), который используется в динамических библиотеках.

```
1   root@kali:~/Desktop/linuxLib/Dyn# make
2   gcc -c main.c
3   gcc -c -fPIC fileRead.c
4   gcc -c -fPIC fileWrite.c
5   gcc -shared -o libMyLib.so fileRead.o fileWrite.o
```

```
gcc -o binary main.o -L. -lMyLib -Wl,-rpath,.
root@kali:~/Desktop/linuxLib/Dyn# ./binary
123456789
23456789
3456789
456789
56789
6789
789
89
9
root@kali:~/Desktop/linuxLib/Dyn# ls -l
total 56
-rwxr-xr-x 1 root root 8448 Dec 10 04:10 binary
-rw------- 1 root root  216 Dec  9 14:58 fileRead.c
-rw-r--r-- 1 root root 1776 Dec 10 04:10 fileRead.o
-rw------- 1 root root  208 Dec 10 03:45 fileWrite.c
-rw-r--r-- 1 root root 1776 Dec 10 04:10 fileWrite.o
-rwxr-xr-x 1 root root 8152 Dec 10 04:10 libMyLib.so
-rw------- 1 root root   67 Dec  9 14:39 main.c
-rw-r--r-- 1 root root 1456 Dec 10 04:10 main.o
-rw------- 1 root root  358 Dec  9 14:56 makefile
-rw------- 1 root root   53 Dec  6 14:25 myLib.h
-rw-r--r-- 1 root root   54 Dec 10 04:10 tempFile.txt
```

Листинг 4.6: Компиляция и запуск

Программа успешно скомпилировалась и отработала. Конечный размер программы составил 8448 байт.

4.3 Статическая библиотека

Немного изменим makefile.

```
# Makefile for myStaticLib project

binary: main.o libworld.a
    gcc -o binary main.o -L. -lworld

main.o: main.c
    gcc -c main.c

libworld.a: fileRead.o fileWrite.o
    ar cr libworld.a fileRead.o fileWrite.o

fileRead.o: fileRead.c
    gcc -c fileRead.c

fileWrite.o: fileWrite.c
    gcc -c fileWrite.c

clean:
    rm -f *.o *.a binary
```

Листинг 4.7: makefile

Команда ar создает библиотеку (архив). В нашем случае два объектных файла объединяются в один файл. Опция -l, переданная компилятору, обрабатывается и посылается

линковщику для того, чтобы тот подключил к бинарнику библиотеку. Опция -L указывает линковщику, где ему искать библиотеку.

```
root@kali:~/Desktop/linuxLib/Stat# make
gcc -c main.c
gcc -c fileRead.c
gcc -c fileWrite.c
ar cr libMyLib.a fileRead.o fileWrite.o
gcc -o binary main.o -L. -lMyLib
root@kali:~/Desktop/linuxLib/Stat# ./binary
123456789
23456789
3456789
456789
56789
6789
789
89
9
root@kali:~/Desktop/linuxLib/Stat# ls -l
total 52
-rwxr-xr-x 1 root root 8744 Dec 10 04:17 binary
-rw------- 1 root root  216 Dec  9 14:58 fileRead.c
-rw-r--r-- 1 root root 1776 Dec 10 04:17 fileRead.o
-rw------- 1 root root  208 Dec 10 03:45 fileWrite.c
-rw-r--r-- 1 root root 1776 Dec 10 04:17 fileWrite.o
-rw-r--r-- 1 root root 3772 Dec 10 04:17 libMyLib.a
-rw------- 1 root root   67 Dec  9 14:39 main.c
-rw-r--r-- 1 root root 1456 Dec 10 04:17 main.o
-rw------- 1 root root  323 Dec 10 04:07 makefile
-rw------- 1 root root   53 Dec  6 14:25 myLib.h
-rw-r--r-- 1 root root   54 Dec 10 04:17 tempFile.txt
```

Листинг 4.8: Компиляция и запуск

Программа успешно скомпилировалась и отработала. Конечный размер программы составил 8744 байт.

Вывод

В данной работе были исследованы статические и динамические библиотеки в операционной системе windows и linux.

Реализация для windows потребовала больше времени, по большей части это связано с тем что работа происходила с использованием Visual Studio, где необходимо настроить множество параметров. А в linux для этого был создан makefile в котором и были выставлены необходимые параметры.

Как и ожидалось для обеих платформ, программа с динамической библиотекой была меньше по размеру, чем программа со статической библиотекой.

Литература

[1] dllexport, dllimport functions [Электронный ресурс]. — URL: https://msdn.microsoft.com/ru-ru/library/3y1sfaz2.aspx (дата обращения: 2017-12-06).

[2] Создание и использование статической библиотеки [Электронный ресурс]. — URL: https://msdn.microsoft.com/ru-ru/library/ms235627.aspx (дата обращения: 2017-12-06).

[3] GetFileTime function [Электронный ресурс]. — URL: https://msdn.microsoft.com/ru-ru/library/windows/desktop/ms724320(v=vs.85).aspx (дата обращения: 2017-12-06).

[4] Создание и использование библиотеки DLL [Электронный ресурс]. — URL: https://msdn.microsoft.com/ru-ru/library/ms235636.aspx (дата обращения: 2017-12-06).

[5] gcc -L / -l option flags [Электронный ресурс]. — URL: https://www.rapidtables.com/code/linux/gcc/gcc-l.html (дата обращения: 2017-12-06).

[6] Shared libraries with GCC on Linux [Электронный ресурс]. — URL: https://www.cprogramming.com/tutorial/shared-libraries-linux-gcc.html (дата обращения: 2017-12-06).

[7] Динамические и статические библиотеки [Электронный ресурс]. — URL: https://www.opennet.ru/docs/RUS/zlp/003.html (дата обращения: 2017-12-06).

[8] Shared Libraries [Электронный ресурс]. — URL: http://tldp.org/HOWTO/Program-Library-HOWTO/shared-libraries.html (дата обращения: 2017-12-06).