МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 7**

по дисциплине:” Системное программирование”

на тему: “Командный язык и скрипты Shell”

Выполнил**:** студент группы 10701321

Мацкевич М.Ю.

Принял**:** Кондратёнок Е.В.

Минск 2023

# Лабораторная работа № 7. Создание и использование библиотек в Linux

# Цель работы: Изучить и закрепить на практике создание и использование статически и динамически подключаемых библиотек в операционных системах семейства Linux

**Задание 1**

Произвести рефакторинг проекта предыдущей лабораторной работы поавтосборке проекта: вынесите код функций бизнес логики в отдельнуюстатическую библиотеку. Для автосборки проекта используйте утилитуmake, обязательно.

При помощи программы ldd проверим зависимости, образовавшиеся междубиблиотеками.Сравните сумму размеров объектных файлов проекта и исполняемого файла.Результаты отобразите скриншотами.

Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет результат?

**Решение**

Напишем makefile, который будет собирать нашу программу

CC = g++

CFLAGS = -Wall -std=c++11

# Исходные файлы

SRCS = program.cpp

# Имя статической библиотеки

LIB = libprogram.a

# Имя исполняемого файла

EXEC = guess\_number\_game

all: $(EXEC)

$(EXEC): main.cpp $(LIB)

$(CC) $(CFLAGS) -o $@ $< -L. -lprogram

$(LIB): $(SRCS)

$(CC) $(CFLAGS) -c $^

ar rcs $@ $(SRCS)

.PHONY: clean

clean:

rm -f $(EXEC) $(LIB)

Имеется исходная структура нашей программы (рис.1)

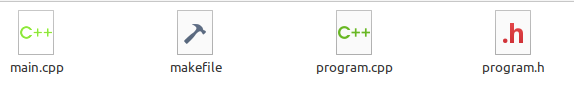


Рис.1 Исходная структура программы

Выполним команду make (рис.2) и получим новые файлы (рис.3)

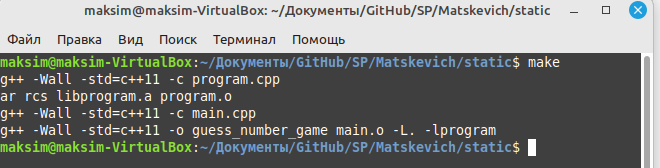


Рис.2 – Компиляция программы



Рис.3 Новая структура программы

Был создан исполняемый файл, внутри которого заложена библиотека, поэтому перемещая файл, он продолжит работать. Если посмотреть зависимости через ldd, то там будет указаны только стандартные библиотеки (рис.4)

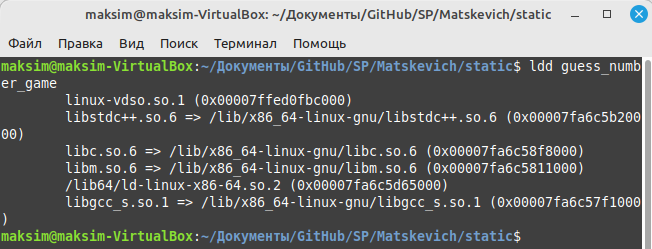


Рис.4 Зависимости

**Задание 2**

Произвести рефакторинг проекта предыдущей лабораторной работы по автосборке проекта: вынесите код функций бизнес логики в отдельную динамическую библиотеку со статическим вызовом. Для автосборки проекта используйте утилиту make, обязательно.

При помощи программы ldd проверим зависимости, образовавшиеся между библиотеками.

Сравните сумму размеров объектных файлов проекта и исполняемого файла.

Результаты отобразите скриншотами.

Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет результат? Если возникла проблема, решите ее и поясните каким способом.

Сравните результаты первого и вторгог упражнения, сделайте вывод.

**Решение**

Для выполнения задания напишем скрипт, который будет собирать библиотеку

g++ -shared -o libprogram.so -fPIC program.cpp

Соберём библиотеку. Результат представлен на рисунке 5.



Рис.5 – файлы после создания библиотеки

Теперь необходимо перенести библиотеку в папку /usr/lib/

Напишем makeftile и соберём программу (рис.6)

CC = g++

CFLAGS = -Wall -std=c++11

# Исходные файлы

SRCS = main.cpp

# Имя исполняемого файла

EXEC = guess\_number\_game

# Имя динамической библиотеки

LIB = libprogram.so

all: $(EXEC)

$(EXEC): $(SRCS) $(LIB)

$(CC) $(CFLAGS) -o $@ $(SRCS) -L. -lprogram

.PHONY: clean

clean:

rm -f $(EXEC)



Рис.6. Файлы после сборки программы

В данном приложении, уже образовалась связь между нашей библиотекой и программой(рис.7)

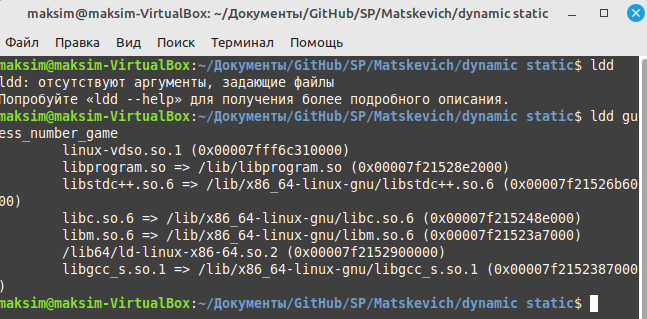


Рис. 7 Завсисмости

Т.к. наша программа связана с библиотекой из папки lib, то если мы перенесём исполняемый файл, то программа продолжит работать корректно.

**Задание 3**

Произвести рефакторинг проекта предыдущей лабораторной работы по автосборке проекта: вынесите код функций бизнес логики в отдельную динамическую библиотеку с динамическим вызовом. Для автосборки проект используйте утилиту make, обязательно.

При помощи программы ldd проверим зависимости, образовавшиеся междубиблиотеками. Сравните сумму размеров объектных файлов проекта и исполняемого файла.Результаты отобразите скриншотами.

Копию исполняемого файла расположите на «Рабочем столе», запустите про-грамму. Какой будет результат? Если возникла проблема, решите ее и поясните каким способом

**Решение**

Для решения опять создадим скрипт для сборки библиотеки и выполним его(рис.8)

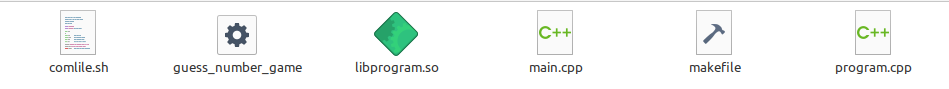


Рис.8 Файлы после сборки библиотеки

Нужно внести изменения в main файл, для того, чтобы библиотека подключалась  
oid\* lib = dlopen("/home/maksim/Документы/GitHub/SP/Matskevich/dynamic dynamic/libprogram.so", RTLD\_LAZY);

if (!lib) {

std::cerr << "Ошибка загрузки библиотеки: " << dlerror() << std::endl;

return 1;

}

typedef void (\*PlayGameFunc)(int, int, int);

PlayGameFunc playGame = reinterpret\_cast<PlayGameFunc>(dlsym(lib, "\_Z8playGameiii"));

if (!playGame) {

std::cerr << "Ошибка загрузки функции: " << dlerror() << std::endl;

dlclose(lib);

return 1;

}

Теперь можно собрать библиотеку через make(рис.9)



Рис.9 – файлы после сборки программы

В зависимостях теперь нет библиотеки, потому-что она подключается по время выполнения (рис. 10)

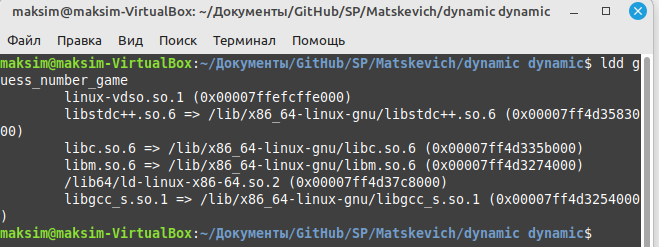


Рис.10 – Зависимости приложения

Если мы переместим файл на рабочий стол, то всё будет работать, потому что мы подключаем библиотеку внтури программы, и указан полный путь к файлу, однако если мы переместим библиотеку, то программа перестанет работать.

Вывод*:* В ходе выполнения лабораторной работы научились работать с библиотеками.