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 № 8**

по дисциплине: ”Системное программирование”

на тему: ***”*** **Создание и использование библиотек в Linux*”***

Выполнил**:** студент группы *10702121* Агеенко А. К.

Высоцкий М. Л.

Градусов А. Р.

Бродко Г. Г.

Принял**:** пр. Давыденко Н. В.

Минск 2023

# Лабораторная работа №8.

**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 по автосборке проекта: вынесите код функций бизнес логики в отдельную статическую библиотеку. Для автосборки проекта используйте утилиту make, обязательно.

При помощи программы ldd проверим зависимости, образовавшиеся между библиотеками.

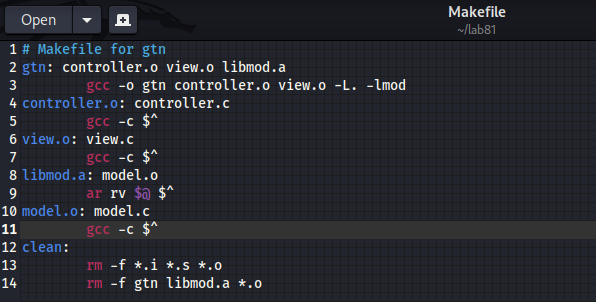
Сравните сумму размеров объектных файлов проекта и исполняемого файла.

Результаты отобразите скриншотами.

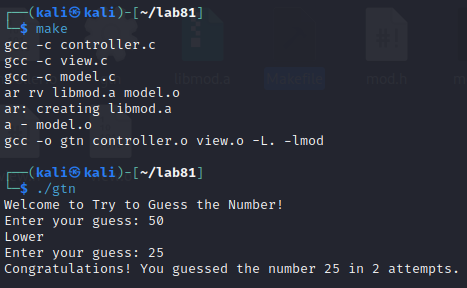
Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет результат?

Решение

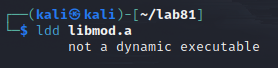
**Создаем Makefile.**



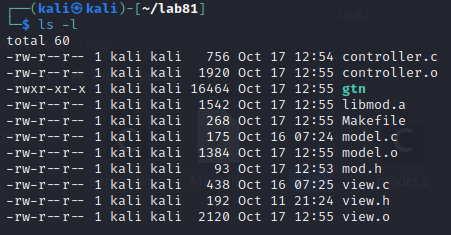
**При помощи утилиты make выполняем автосборку статической библиотеки и проекта и проверяем работоспособность.**



**При помощи программы ldd проверим зависимости, образовавшиеся между библиотеками.**

 мы не можем это сделать, так как у нас статическая библиотека.

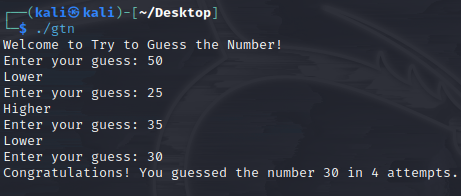
**Сравниваем сумму размеров объектных файлов проекта и исполняемого файла.**



Сумма объёма объектных файлов будет меньше(4.4кб), чем исполняемый файл(16.1кб).

Скорее всего, это за счет наличия разного рода управляющей информации, определения адресов расположения присоединенных к нему функций и т.д.

**Располагаем исполняемый файл на рабочем столе и запускаем программу.**



Программа запустилась успешно!

## 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.

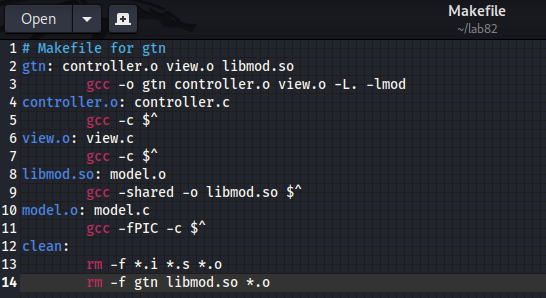
Сравните сумму размеров объектных файлов проекта и исполняемого файла.

Результаты отобразите скриншотами.

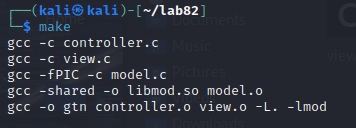
Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет результат? Если возникла проблема, решите ее и поясните каким способом.

Сравните результаты первого и вторгог упражнения, сделайте вывод.

**Создаем Makefile.**



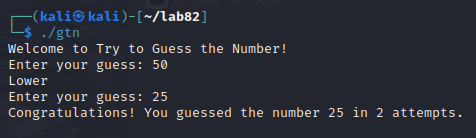
**При помощи утилиты make выполняем автосборку динамической библиотеки и проекта.**



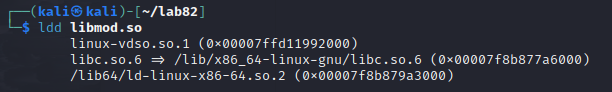
**Устанавливаем переменную LD\_LIBRARY\_PATH и заносим туда абсолютное имя.**



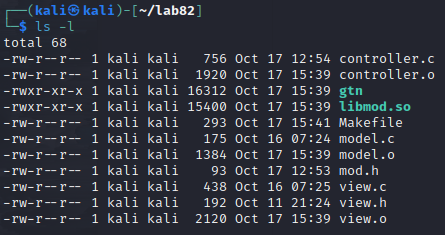
**проверяем работоспособность.**



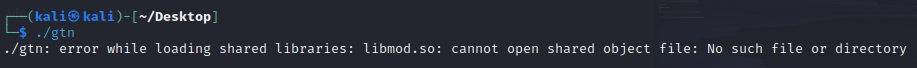
**При помощи программы ldd проверим зависимости, образовавшиеся между библиотеками.**



**Сравните сумму размеров объектных файлов проекта и исполняемого файла.**



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



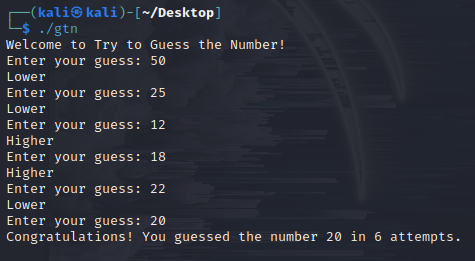
В текущем каталоге не найдено данной библиотеки.

Решаем проблему:



Указываем переменной LD\_LIBRARY\_PATH путь к библиотеки.

запускаем исполняемый файл



**Сравниваем результаты первого и второго упражнения, делаем вывод.**

Процесс создания и подключения совместно используемых библиотек несколько сложнее, чем статических

## 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 проекта используйте утилиту make, обязательно.

При помощи программы ldd проверим зависимости, образовавшиеся между библиотеками.

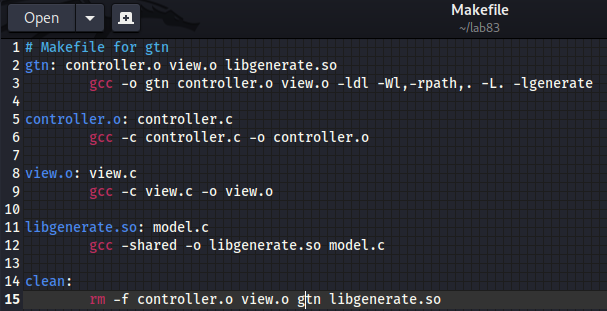
Сравните сумму размеров объектных файлов проекта и исполняемого файла.

Результаты отобразите скриншотами.

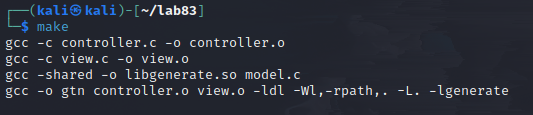
Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет результат? Если возникла проблема, решите ее и поясните каким способом.

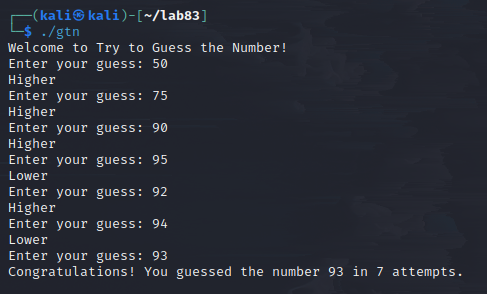
Сравните результаты с предыдущими упражнения, сделайте вывод

**Создаем Makefile.**

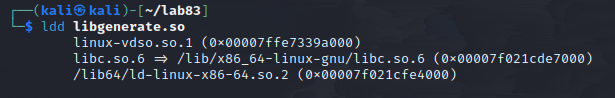


**При помощи утилиты make выполняем автосборку динамической библиотеки и проекта.**

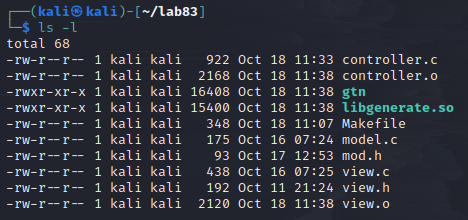




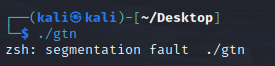
**При помощи программы ldd проверим зависимости, образовавшиеся между библиотеками.**



**Сравните сумму размеров объектных файлов проекта и исполняемого файла.**



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



Эта ошибка возникает, когда программа пытается обратиться к памяти, к которой у нее нет доступа.



устанавливаем переменную окружения **LD\_LIBRARY\_PATH** в соответствующий каталог, где находится библиотека.

**Сравните результаты с предыдущими упражнения, сделайте вывод**

Поскольку весь код в статических библиотеках включается в исполняемый файл, это уменьшает вероятность конфликтов между версиями библиотек и повреждения системных библиотек.

Исполняемые файлы, созданные с использованием статических библиотек, могут быть крупными из-за включенного кода, что занимает дополнительное дисковое пространство.

Если динамическая библиотека требует обновления, достаточно заменить её на более новую версию без перекомпиляции программы.

1) **Что такое библиотеки и для чего они используются?**

Библиотеки позволяют разным программам использовать один и тот же объектный код. Библиотека (library) – это набор соединенных определенным образом объектных файлов. Библиотеки подразделяются на две категории: − статические (архивы); − динамические (совместно используемые)

2) **Чем отличается заголовочные файлы (файлы с расширением \*.h) от файлов библиотек?**

- Заголовочные файлы содержат объявления функций, структур, классов и переменных, которые могут быть использованы в других частях программы. Они обычно содержат только сигнатуры (прототипы) функций и объявления данных без их фактической реализации.

- Файлы библиотек, как статические (.a) или динамические (.so или .dll), содержат скомпилированный исполняемый код, который может быть использован другими программами. Они включают в себя реализации функций и данные, которые экспортируются для использования в других программах.

3) **Опишите существующие типы библиотек и принципы их использования.**

**- Статические библиотеки (.a или .lib):** Эти библиотеки компилируются непосредственно в исполняемый файл во время компиляции. Принцип использования заключается в компиляции исходного кода с этими библиотеками. Преимущества: независимость от наличия внешних библиотек при запуске программы. Недостатки: увеличение размера исполняемого файла и необходимость перекомпиляции при обновлении библиотеки.

**- Динамические библиотеки (.so):** Эти библиотеки загружаются в память во время выполнения программы. Принцип использования заключается в подключении к ним при выполнении. Преимущества: экономия памяти, обновление библиотеки без перекомпиляции программы. Недостатки: зависимость от наличия библиотеки при выполнении, небольшая дополнительная накладная работа на загрузку и связывание.

4) **Опишите преимущества и недостатки каждого из типа библиотек.**

**Статические библиотеки:**

* + Преимущества: Независимость от наличия внешних библиотек при запуске программы, отсутствие дополнительных файлов при выполнении, более простое развертывание.
  + Недостатки: Увеличение размера исполняемого файла, необходимость перекомпиляции при обновлении библиотеки.

**Динамические библиотеки:**

* + Преимущества: Экономия памяти, обновление библиотеки без перекомпиляции программы, возможность одновременного использования нескольких версий библиотеки.

Недостатки: Зависимость от наличия библиотеки при выполнении, небольшая дополнительная накладная работа на загрузку и связывание, сложнее управление зависимостями.

5) **Как и какими способами подключить библиотеку к программе? Опишите специфику подключения.**

Подключение библиотеки к программе зависит от типа библиотеки (статической или динамической) и используемого компилятора. Вот общие способы и специфика их использования:

**Подключение статических библиотек:**

**1. Подключение библиотеки при компиляции:**

1. Спецификация компилятора: Вы указываете имя статической библиотеки с флагом **-l**, а путь к библиотеке с флагом **-L**.h/t

o/library -lmylibrary

2. Указание полного пути к библиотеке: Можно указать полный путь к статической библиотеке без использования флага **-l**.ry/libmylibrary.a

В этом случае код библиотеки включается в исполняемый файл, и программу можно запускать независимо от наличия библиотеки на системе.

1. **Подключение библиотеки в коде:**

Вы можете явно указать файлы объектного кода библиотеки в командной строке компиляции:

В этом случае объектные файлы из библиотеки станут частью исполняемого файла.

**Подключение динамических библиотек:**

**2. Подключение библиотеки при компиляции:**

* + Спецификация компилятора: Вы указываете имя динамической библиотеки с флагом **-l**, а путь к библиотеке с флагом **-L**. При выполнении программы, операционная система автоматически ищет и загружает динамические библиотеки.

1. **Использование переменных окружения:**

В Linux и Unix-подобных системах можно использовать переменную окружения **LD\_LIBRARY\_PATH**, чтобы указать пути к директориям, в которых операционная система будет искать динамические библиотеки. Например:

Этот способ удобен, но используется временно.

1. **Использование флагов линковки:**

При компиляции программы, вы можете использовать флаги линковки для указания пути к библиотеке:/to/library

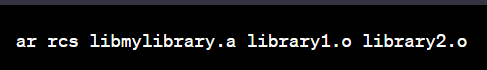
6) **На какой стадии происходит подключение библиотек (объектных файлов) к основному модулю программы?**

Подключение библиотек (как статических, так и динамических) к основному модулю программы происходит на этапе компиляции и/или линковки программы.

7) **Как создать статическую и динамическую библиотеки и подключить их к программе****?**

**1. Создание статической библиотеки:**

**Создание архива:** С помощью утилиты **ar** создайте архив статической библиотеки



**Использование статической библиотеки в программе:** При компиляции программы укажите путь к статической библиотеке и имя библиотеки:



**Создание и компиляция исходных файлов:** Создайте исходные файлы библиотеки, например, **.c .** Затем компилируйте их в объектные файлы с использованием компилятора, например:



**2. Создание динамической библиотеки:**

**Компиляция в объектные файлы:** Компилируйте исходные файлы в объектные файлы, но при этом используйте флаг **-fPIC** (position-independent code), который необходим для создания динамических библиотек:



**Создание динамической библиотеки:** С помощью компилятора создайте динамическую библиотеку:



**Использование динамической библиотеки в программе:** При компиляции программы укажите путь к динамической библиотеке и имя библиотеки:



8) **Что такое «позиционно-независимый код» (position independent code, PIC)?**

PIC - это код, который может быть загружен в память и выполняться независимо от его фактического расположения в памяти. Это важно для динамических библиотек, так как они загружаются в произвольное место памяти. Позиционно-независимый код имеет возможность подгружаться к программе в момент ее запуска. Чтобы получить объектный файл с позиционнонезависимым кодом, нужно откомпилировать исходный файл с опцией -fPIC

9) **Какие способы можно использовать для того, чтобы указать местоположение подключаемых динамических библиотек?**

1) Используйте переменные окружения, такие как LD\_LIBRARY\_PATH чтобы указать пути к директориям, в которых операционная система будет искать динамические библиотеки.

2) Использование опции -rpath при компиляции: При компиляции программы вы можете использовать опцию **-rpath** для указания пути, где операционная система будет искать динамические библиотеки при выполнении программы.

10) **Если в целевом каталоге присутствует два типа одной и той же библиотеки, какая из низ всегда будет подключаться по умолчанию? А как явно подключить вторую?**

По умолчанию будет подключаться динамическая библиотека, чтобы подключить вторую - необходимо указать флаг -static.

11) **Какие проблемы могут быть при обновлении совместно используемых библиотек, и что такое «Ад DLL» (DLL HELL).**

Проблемы, связанные с обновлением совместно используемых библиотек, могут включать в себя следующие аспекты:

1. **Совместимость обратной совместимости:** Если новая версия библиотеки не обеспечивает полную обратную совместимость с предыдущими версиями, это может привести к тому, что программы, зависящие от старой версии, могут перестать работать или работать неправильно после обновления.
2. **Зависимости от конкретной версии:** Некоторые программы могут быть жестко привязаны к конкретной версии библиотеки, и обновление этой библиотеки может вызвать конфликты и ошибки.
3. **Совместное использование библиотек между разными программами:** Если несколько программ зависят от одной и той же библиотеки, обновление этой библиотеки может повлиять на все эти программы.
4. **Параллельное существование нескольких версий:** В некоторых случаях может потребоваться, чтобы несколько версий одной и той же библиотеки сосуществовали на одной системе. Например, разные программы могут требовать разные версии библиотеки.

"Ад DLL" (DLL Hell) - это термин, используемый для описания сложностей и проблем, связанных с управлением совместно используемыми библиотеками, особенно в операционных системах Windows.