МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

О Т Ч Е Т

ПО ЛАБОРАТОРНОЙ РАБОТЕ №**4**

«Создание и использование библиотек в Linux»

по курсу: "Системное программирование"

Выполнили: Бартось П.Д.

Кабыш Я.А.

Хитрик Д.С.

Проверил: Станкевич С.Н.

**Цель работы :**

Изучить и закрепить на практике создание и использование статически и динамически подключаемых библиотек в операционных системах семейства Linux.

**Требования :**

1) Разработать модульное консольное приложение на С/С++ согласно варианту задания. Для компиляции, компоновки и выполнения программы использовать стандартный компилятор Linux gcc/g++, а для отладки - gdb.

2) Размерность массива задаётся пользователем на стадии выполнения программы. Для повышения производительности программы необходимо использовать динамическое выделение памяти.

3) Во время работы программы должны выводить на экран исходные и конечные данные.

4) При выполнении задания запрещается использовать интегрированные средства разработки (Integrated Development Environment, IDE). Рекомендуется задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++.

5) При разработке программ необходимо придерживаться соглашения по написанию кода на C/C++ (Code-Convention).

**Задания:**

*Задание 1:*

Скопилируем библиотеку и на выходе получим объектный файл нашей библиотеки.



Создадим статическую библиотеку с помощью ar(r - предписывает заменять старые версии объектных файлов новыми - необходим для переупаковки библиотеки; c - создать статическую библиотеку, если та еще не существует).



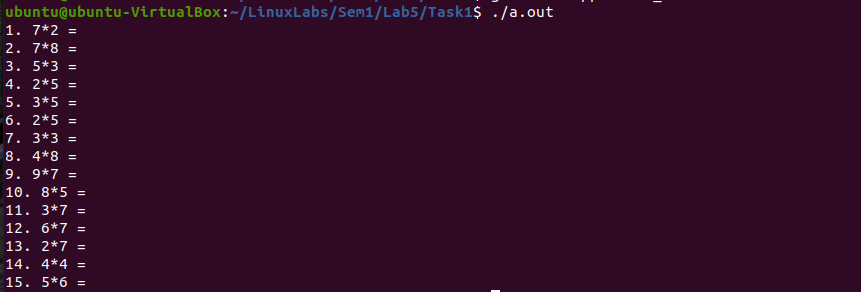
Проиндексируем функции внутри библиотеки для более быстрой линковки:



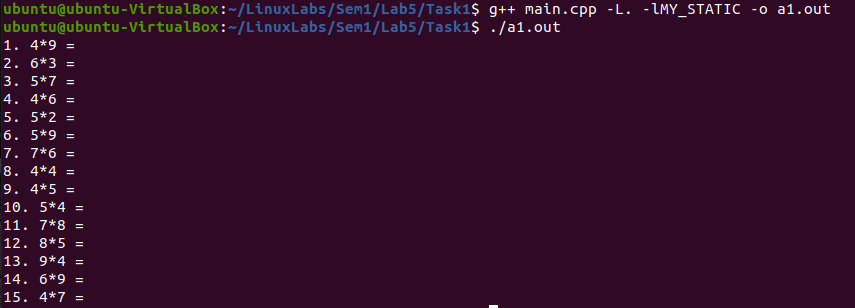
Скомпилируем и слинкуем (в том числе с нашей библиотекой) нашу программу:



И запустим:



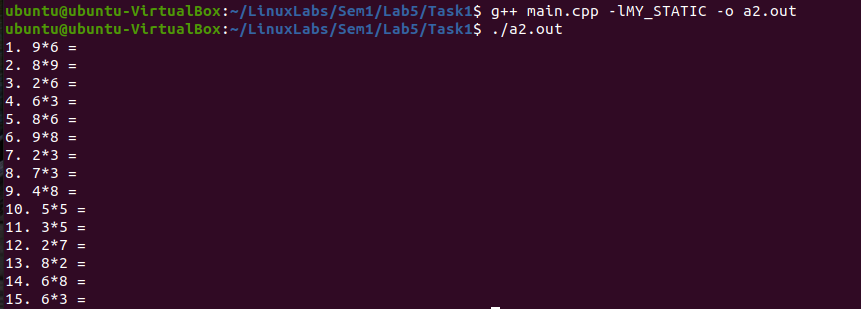
Скомпилируем с другой командой и получим:



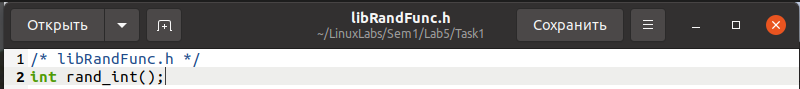
Занесем библиотеку в /usr/lib:



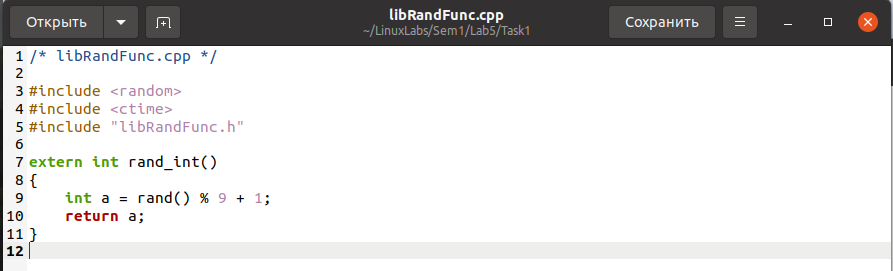
И снова скомпилируем и запустим программу:



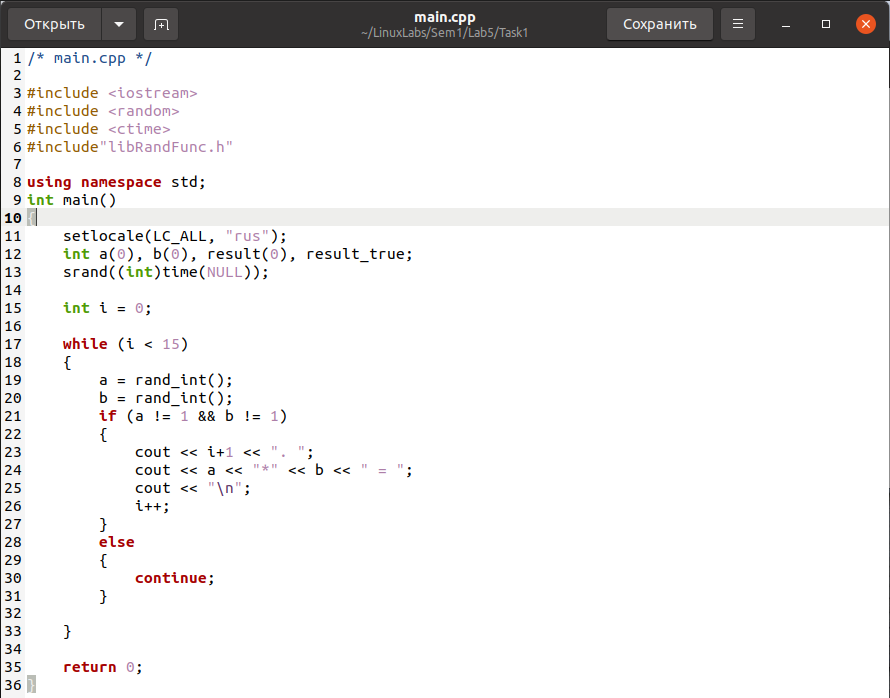
Заголовочный файл с прототипом библиотечной функции:



Содержание файла с функцией:

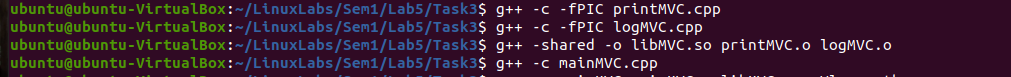


Содержание главного файла:

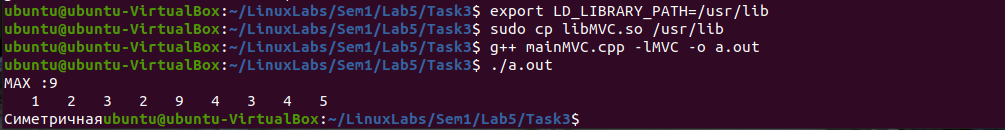


*Задание 2:*

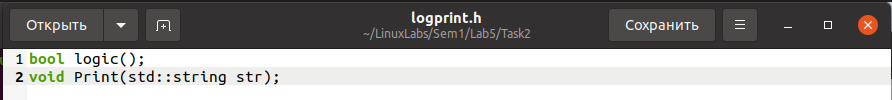
Создаем объектные файлы, и из них динамическую библиотеку:



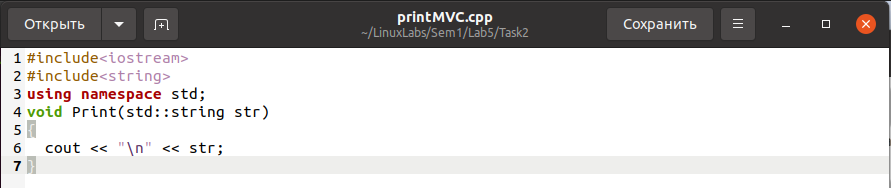
Копируем библиотеку в /usr/lib, прописываем путь LD\_LIBRARY\_PATH и компилируем программу и линкуем ее с библиотекой, получаем результат:

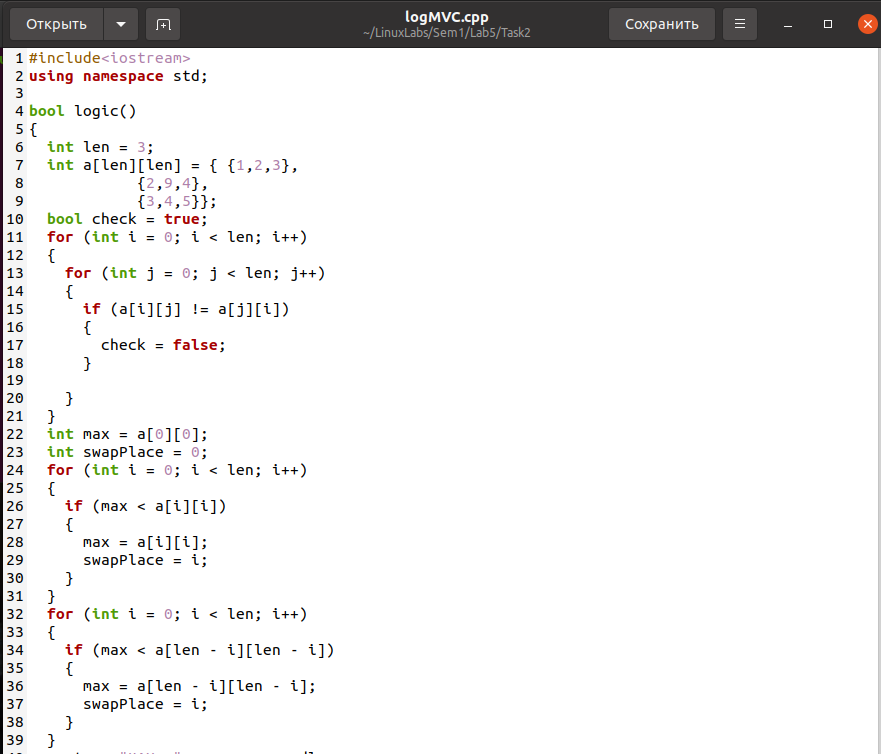


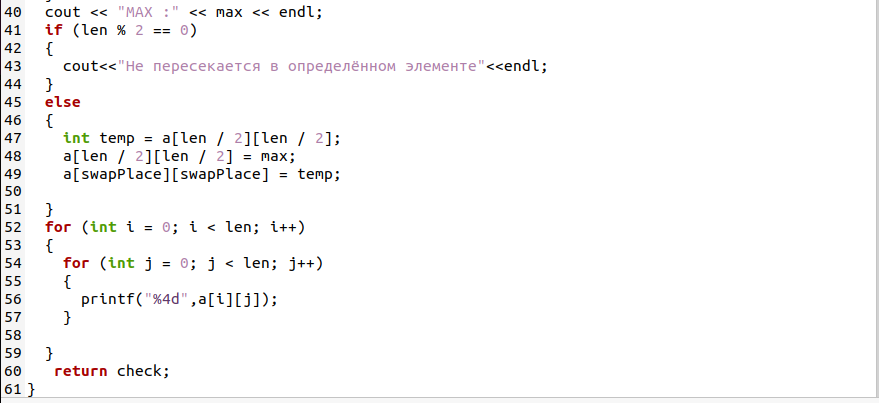
Заголовочный файл с прототипами библиотечный функции:



Файлы с определениями функций:







Содержание главного файла:



*Задание3:*

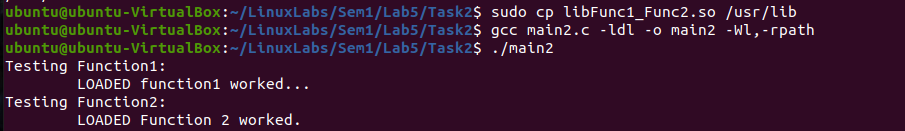
Создаём объектные файлы:



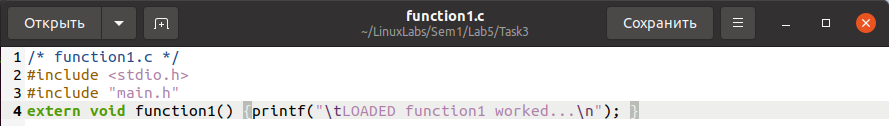
Из объектных файлов создаём динамическую библиотеку:

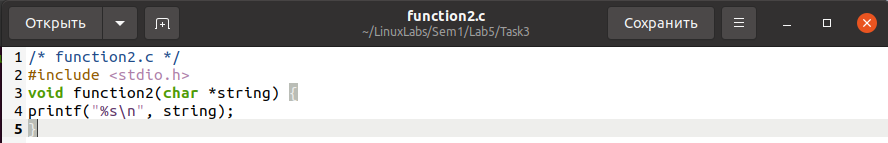


Копируем библиотеку в /usr/lib, собираем динамическую библиотеку подключаемую динамическим способом с главным файлом и получаем результат:

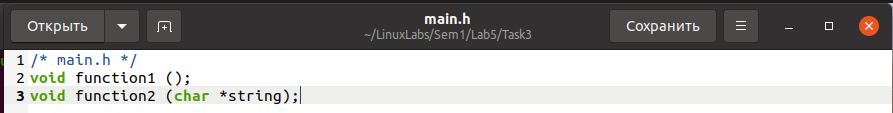


Содержание файлов с функциями:

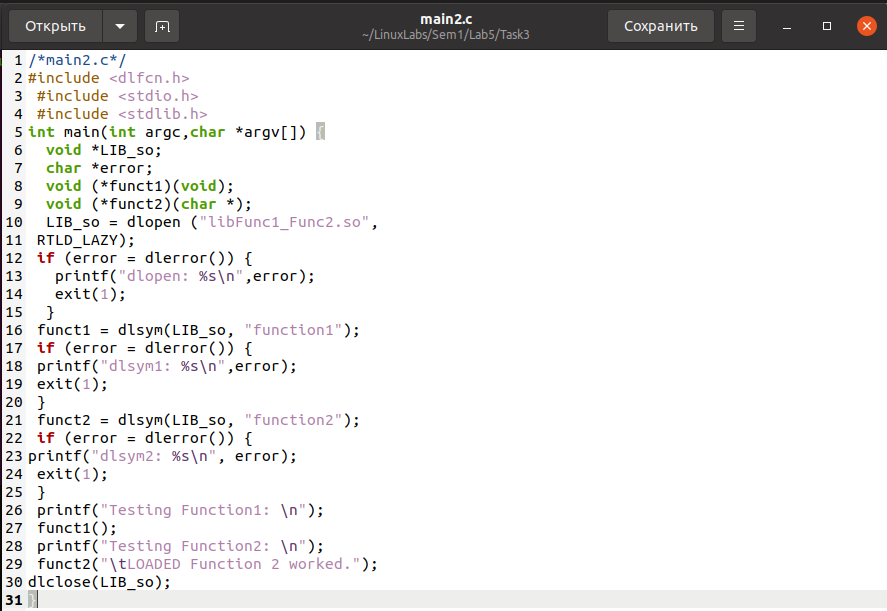




Содержание заголовочного файла:



Код главного файла:



**Контрольные вопросы:**

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

Библиотека (library) – это набор соединенных определенным образом объектных файлов. Библиотеки позволяют разным программам использовать один и тот же объектный код.

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

Иногда заголовочные файлы тоже называют библиотеками. Это не так! Библиотеки – это объектный код, сгруппированный таким образом, чтобы им могли пользоваться разные программы. Заголовочные файлы – это часть исходного кода программы, в которых, как правило, определяются соглашения по использованию общих идентификаторов (имен). Когда в заголовочном файле определены механизмы, реализованные в библиотеке, то правильно будет называть такой файл (или группу файлов) интерфейсом библиотеки.

1. **Опишите существующие типы библиотек и принципы их использования.**

Статическая библиотека – это архив, создаваемый специальным архиватором ar из пакета GNU binutils. Утилита ar создает архив, который может подключаться к программе во время компоновки на правах библиотеки. Файлы статических библиотек имеют расширение .a.

*Совместно используемые библиотеки*, их также можно называть *динамическими*, не помещают свой код непосредственно в программу, а лишь создают специальные ссылки. Поэтому любая программа, скомпонованная с динамической библиотекой, при запуске требует наличия данной библиотеки в системе. Совместно используемые библиотеки создаются компоновщиком при вызове gcc с опцией -shared и имеют расширение .so

1. **Опишите преимущества и недостатки каждого из типа библиотек.**

Познакомившись со статическими архивами и совместно используемыми библиотеками. читатели, очевидно, задумались: какие же из них лучше использовать? Есть несколько важных моментов, о которых следует помнить.

Большим преимуществом совместно используемой библиотеки является то. что она экономит место на диске при инсталляции программы. Когда устанавливаются десять программ и все они работают с одной и той же библиотекой, экономия может оказаться весьма существенной, тогда как статический архив будет включен во все десять программ. Уменьшается также время загрузки, если программа загружается из Internet.

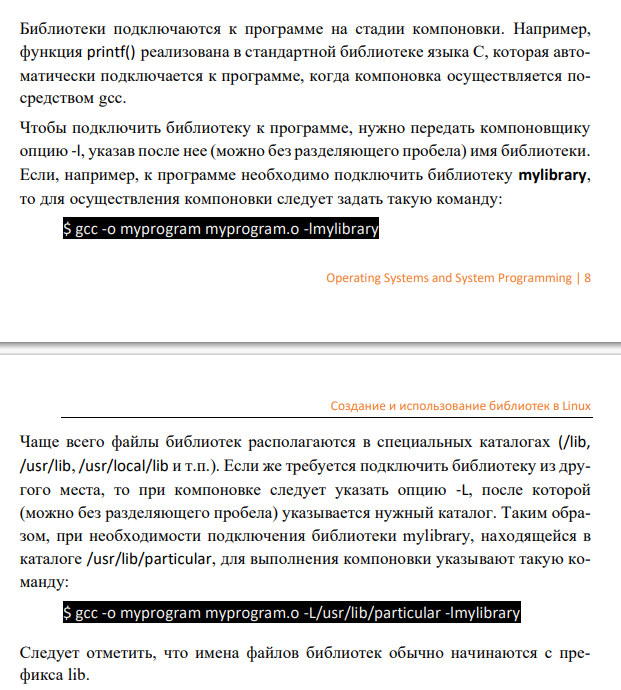
С этим связано еще одно преимущество совместно используемых библиотек: пользователи могут обновлять библиотеки, не затрагивая связанные с ними программы. Предположим, к примеру, что была создана библиотека, содержащая функции управления HTTP- соединениями. Потенциально с ней может работать множество программ. Если впоследствии в библиотеке обнаружится ошибка, достаточно будет просто заменить ее файл, и все программы, использующие данную библиотеку, немедленно обновятся. Не придется выполнять перекомпоновку всех программ, как в случае статического архива.

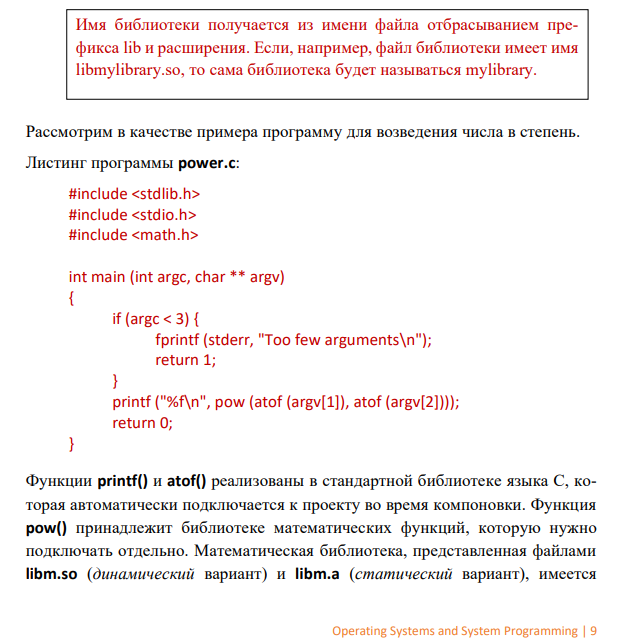
Описанные преимущества могут заставить читателей подумать, будто статические архивы бесполезны. Но их существование обусловлено вескими причинами. Тот факт, что обновление совместно используемой библиотеки отражается на всех связанных с нею программах, может на самом деле оказаться недостатком. Некоторые программы тесно связаны с используемыми библиотеками и не должны зависеть от произвольных изменений в системе.

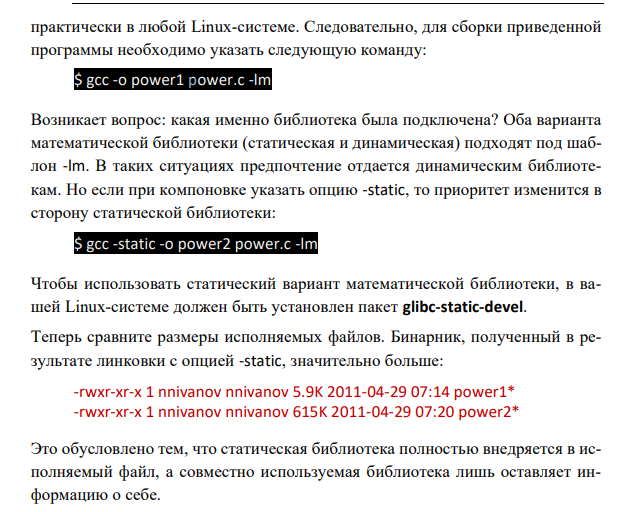
Если библиотеки не должны инсталлироваться в каталог /lib или /usr/lib, нужно дважды подумать, стоит ли их делать совместно используемыми. (Библиотеки нельзя помещать в указанные каталоги, если предполагается, что программу будут инсталлировать пользователи, не имеющие привилегий системного администратора.) В частности, прием с флагом -Wl,-rpath не будет работать, поскольку не известно, где именно окажутся библиотеки. А просить пользователей устанавливать переменную LD\_LIBRARY\_PATH — не выход из положения, так как это означает для них выполнение дополнительного (для некоторых — не самого тривиального) действия.

Оценивать преимущества и недостатки двух типов библиотек нужно для каждой создаваемой программы отдельно.

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

****

****

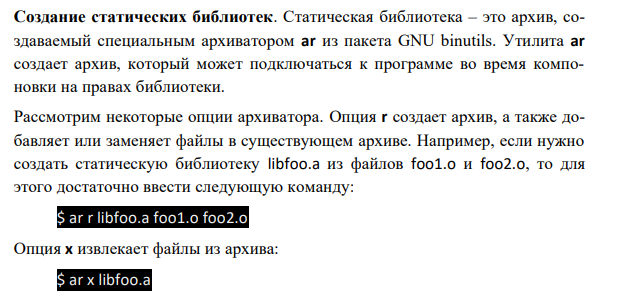
****

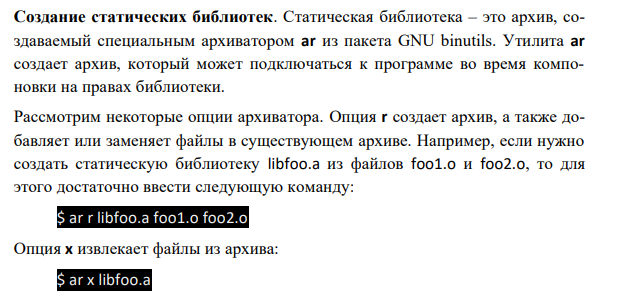
1. **На какой стадии происходит подключение библиотек (объектных файлов) к основному модулю программы?**

Динамический библиотеки могут вызываться (подгружаться) двумя способами: статическим и динамическим. Статический способ подгрузки библиотеки мы рассмотрели выше. Здесь мы рассмотрим динамический вызов динамической библиотеки. Этот способ вызова в некотором роде проще чем статический вызов. Суть динамической подгрузки состоит в том, что запущенная программа может по собственному усмотрению подключить к себе какую-либо (динамическую) библиотеку. Благодаря этой возможности создаются программы с подключаемыми плагинами, такие как XMMS. Для этого используется функция dlopen(). Вызов dlopen(), загружает в память динамическую (разделяемую) библиотеку (если она еще не загружена) и возвращает идентификатор, используемый для адресации к ее функциям.

Вызов dlsym() возвращает адреса функций, которые потом могут вызываться, как будто бы они находятся в главной программе main. Функция dlclose() отсоединяет (detache) главную (текущую) программу от загруженной разделяемой библиотеки (что ОЧЕНЬ удобно, если необходимо минимизировать объем используемой оперативной памяти). Отметим, что если к динамической библиотеке не присоединено больше ни одной программы, то она выгружается из памяти. Функция dlerror() возвращает строку описания ошибки, произошедшей при последнем вызове одной из функций dlopen(), dlclose(), dlsym(). При отсутствии ошибок dlerror() возвращает значение NULL. Второй аргумент вызова функции dlopen() — флаг способа динамической загрузки библиотеки. Он может иметь следующие значения. При значении RTLD\_NOW все функции библиотеки сразу загружаются в память и после этого становятся доступными для вызова. При значении флага RTLD\_LAZY загрузка каждой функции задерживается до тех пор, пока ее имя не будет передано функции dlsym(). Каждое из двух этих значений флага может быть соединено с помощью ключевого слова OR со значением RTLD\_GLOBAL. При этом все внешние вызовы загружаемой динамической библиотеки разрешаются вызовом функций из других динамических библиотек. Последние при этом также загружаются в (оперативную) память компьютера. Практический пример создания и подключения динамической библиотеки динамическим способом представлен в упражнении №2.

1. **Как создать статическую и динамическую библиотеки и подключить их к программе?**

****

****

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

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.

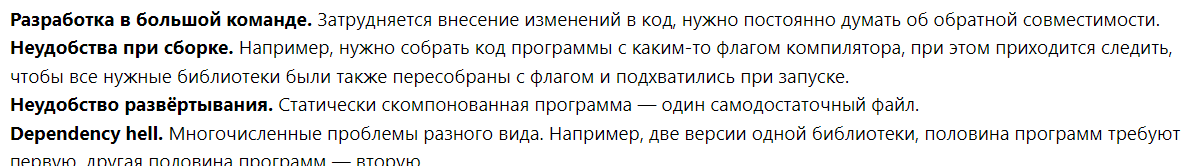
1. **Какие способы можно использовать для того, чтобы указать местоположение подключаемых динамических библиотек?**





**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?**

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

****

**Вывод:** в данной лабораторной работе были изучены встроенный инструментарий для разработки приложений под семейство ОС Linux и этапы такие, как проектирование, компиляция, отладка и запуск.