МИНЕСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Себряковский филиал федерального государственного бюджетного образовательного учреждения высшего образования «Волгоградский государственный технический университет» (Себряковский филиал ФГБОУ ВО ВолгГТУ)

**РЕФЕРАТ**

по дисциплине «ОС Unix»

на тему: «Программирование в Linux»

    Выполнил:

                                                         студент 3 курса группы <Группа>

<ФИО>

       Проверил:

                                                                 Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_

<ФИО>

<Город>, 2021

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc1)

[1. На чем пишут программы? 3](#_Toc2)

[2. Библиотеки 4](#_Toc3)

[3. Процесс сборки программы 5](#_Toc4)

[4. Как собрать программу 6](#_Toc5)

[5. Автоматизация сборки 6](#_Toc6)

[ВЫВОДЫ 7](#_Toc7)

# **ВВЕДЕНИЕ**

[Все действия в операционной системе выполняются с помощью программ, поэтому многим новичкам интересно не только использовать чужие программы, а писать свои. Многие хотят внести свой вклад в кодовую базу OpenSource.](https://juqu1aesh2.semmi.ru/go?block_id=b73e2f33-96bf-4fdb-988b-6588f1e22632&place_id=9045fe0b-1cd4-4f85-afc2-8c4789132283)

Это обзорная статья про программирование под Linux. Мы рассмотрим какие языки используются чаще всего, рассмотрим основные понятия, а также возможности, разберем как написать простейшую программу на одном из самых популярных языков программирования, как ее вручную собрать и запустить.

# **1. На чем пишут программы?**

Исторически сложилось так, что ядро Unix было написано на языке Си. Даже более того, этот язык был создан для написания ядра Unix. Поскольку ядро Linux было основано на ядре Minix (версии Unix), то оно тоже было написано на Си. Поэтому можно сказать, что основной язык программирования для Linux это Си и С++. Такая тенденция сохранялась на протяжении долгого времени.

А вообще, писать программы для Linux можно почти на любом языке начиная от Java и Python и заканчивая С# и даже Pascal. Для всех языков есть компиляторы и интерпретаторы. Писать программы на С++ сложно, а Си многими уже считается устаревшим, поэтому множество программистов используют другие языки для написания программ. Например, множество системных инструментов написаны на Python или Perl. Большинство программ от команды Linux Mint, установщик Ubuntu и некоторые скрипты apt написаны на Python. Множество скриптов, в том числе простые скрипты оптимизации написаны на Perl. Иногда для скриптов используется Ruby. Это скрипты OpenShift или, например, фреймворк Metasploit. Некоторые разработчики кроссплатформенных программ используют Java. Но основные компоненты системы написаны все же на Си.

# **2. Библиотеки**

Естественно, что если вам необходимо вывести строку или изображение на экран, то вы не будете напрямую обращаться к видеокарте. Вы просто вызовете несколько функций, которые уже реализованы в системе и передадите им данные, которые нужно вывести на экран. Такие функции размещаются в библиотеках. Фактически, библиотеки - это наборы функций, которые используются другими программами. В них находится такой же код, как и в других программах, разница лишь в том, там необязательно присутствие функции инициализации.

Библиотеки делятся на два типа:

* Статические - они связываются с программой на этапе компиляции, они связываются и после этого все функции библиотеки доступны в программе как родные. Такие библиотеки имеют расширение .a;
* Динамические - такие библиотеки встречаются намного чаще, они загружены в оперативную память, и связываются с программной динамически. Когда программе нужна какая-либо библиотека, она просто вызывает ее по известному адресу в оперативной памяти. Это позволяет экономить память. Расширение этих библиотек - .so, которое походит от Shared Object.

Таким образом, для любой программы на Си нужно подключать библиотеки, и все программы используют какие-либо библиотеки. Также важно заметить, на каком языке бы вы не надумали писать, в конечном итоге все будет сведено к системным библиотекам Си. Например, вы пишите программу на Python, используете стандартные возможности этого языка, а сам интерпретатор уже является программой на Си/С++, которая использует системные библиотеки для доступа к основным возможностям. Поэтому важно понимать как работают программы на Си. Конечно, есть языки, вроде Go, которые сразу переводятся на ассемблер, но там используются принципы те же, что и здесь. К тому же  системное программирование linux, в основном, это Си или С++.

# **3. Процесс сборки программы**

Каждая серьезная программа состоит из множества файлов, это файлы исходников с расширением .c и заголовочные файлы с расширением .h. Такие заголовочные файлы содержат функции, которые импортируются в программу из библиотек или других файлов .с. Перед тем. как компилятор сможет собрать программу и подготовить ее к работе, ему нужно проверить действительно ли все функции реализованы, доступны ли все статические библиотеки и собрать ее в один файл. Поэтому, первым делом выполняется препроцессор, который собирает исходный файл, выполняются такие инструкции, как include для включения кода заголовочных файлов.

На следующем этапе к работе приступает компилятор, он выполняет все необходимые действия над кодом, разбирает синтаксические конструкции языка, переменные и преобразовывает все это в промежуточный код, а затем в код машинных команд, который мы можем потом посмотреть на языке ассемблера. Программа на этом этапе называется объектный модуль и она еще не готова к выполнению.

Далее к работе приступает компоновщик. Его задача связать объектный модуль со статическими библиотеками и другими объектными модулями. Для каждого исходного файла создается отдельный объектный модуль. Только теперь программа может быть запущена.

# **4. Как собрать программу**

Для сборки программ в Linux используется два типа компиляторов, это Gcc и Clang. Пока что GCC более распространен.

Давайте рассмотрим синтаксис и опции компилятора:

Листинг 1 - Синтаксис команды компилятора

|  |  |
| --- | --- |
| 1 | $ gcc опции исходный\_файл\_1.с -o готовый\_файл |

С помощью опций мы говорим утилите что нужно сделать, какие библиотеки использовать, затем просто указываем исходные файлы программы. Давайте рассмотрим опции, которые будем сегодня использовать:

* -o - записать результат в файл для вывода;
* -c - создать объектный файл;
* -x - указать тип файла;
* -l - загрузить статическую библиотеку.

Собственно, это все самое основное, что понадобится для сборки программы.

# **5. Автоматизация сборки**

Когда мы рассматриваем программирование под Linux невозможно не отметить систему автоматизации сборки программ. Дело в том, что когда исходных файлов программы много, вы не будете вручную вводить команды для их компиляции. Можно записать их один раз, а затем использовать везде. Для этого существует утилита make и файлы Makefile. Этот файл состоит из целей и имеет такой синтаксис:

Листинг 2 - Синтаксис файла Makefile

|  |  |
| --- | --- |
| 1 | цель: зависимости |
| 2 | <Tab> команда |

 В качестве зависимости цели может быть файл или другая цель, основная цель - all, а команда выполняет необходимые действия по сборке.

Затем вам достаточно выполнить команду make для запуска компиляции, только не забудьте удалить предыдущие временные файлы и собранную программу.

# **ВЫВОДЫ**

Создание программ Linux очень интересно и увлекательно. Сложно охватить все в рамках этого реферата, но мы рассмотрели самые основы и они должны дать вам базу. В этой статье мы рассмотрели основы программирования в linux.