Отчет по лабораторной работе 12

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Шалыгин Георгий Эдуардович, НФИбд-02-20

Содержание

# Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

Объект исследования: система UNIX.

Предмет исследования: программирование в UNIX.

# Техническое обеспечение:

* Характеристики техники: AMD Ryzen 5 3500U 2.1 GHz, 8 GB оперативной памяти, 50 GB свободного места на жёстком диске;
* ОС Windows 10 Home
* Git 2.31.1
* Google Chrome 91.0.4472.19
* VirtualBox 2.0
* CentOS 7

# Условные обозначения и термины:

**Текстовым редактором**(text editor) называют программу, которая предназначена для редактирования (составления и изменения) файлов, содержащих только текст. [1]

**Командный язык** - это язык, на котором пользователь взаимодействует с системой в интерактивном режиме.

**Командный интерпретатор**, интерпретатор командной строки - компьютерная программа, часть операционной системы, обеспечивающая базовые возможности управления компьютером посредством интерактивного ввода команд через интерфейс командной строки или последовательного исполнения пакетных командных файлов.[3]

Подробнее в [2] и [3].

# Теоретическое введение:

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

– оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;

– С-оболочка (или csh) — надстройка на оболочкой Борна, использующая Сподобный синтаксис команд с возможностью сохранения истории выполнения команд;

– оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;

– BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linuxподобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже. [3]

**Переменные в языке программирования bash**

Командный процессор bash обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Пользователь имеет возможность присвоить переменной значение некоторой строки символов. Например, команда mark=/usr/andy/bin присваивает значение строки символов /usr/andy/bin переменной mark типа строка символов.

Использование значения, присвоенного некоторой переменной, называется подстановкой. Для того чтобы имя переменной не сливалось с символами, которые могут следовать за ним в командной строке, при подстановке в общем случае используется следующая форма записи: ${имя переменной}. [3]

**Использование арифметических вычислений. Операторы let и read**

Оболочка bash поддерживает встроенные арифметические функции. Команда let является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Простейшее выражение — это единичный терм (term), обычно целочисленный.

Команда let берет два операнда и присваивает их переменной. Положительным моментом команды let можно считать то, что для идентификации переменной ей не нужен знак доллара; вы можете писать команды типа let sum=x+7, и let будет искать переменную x и добавлять к ней 7.

Команда read позволяет читать значения переменных со стандартного ввода:

echo “Please enter Month and Day of Birth ?”

read mon day trash [3]

**Командные файлы и функции**

Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным. Далее этот файл можно выполнить по команде: bash командный\_файл [аргументы]

Чтобы не вводить каждый раз последовательности символов bash, необходимо изменить код защиты этого командного файла, обеспечив доступ к этому файлу по выполнению. Это может быть сделано с помощью команды chmod +x имя\_файла

**Передача параметров в командные файлы и специальные переменные**

Пусть к командному файлу where имеется доступ по выполнению и этот командный файл содержит следующий конвейер: who | grep $1.

В ходе интерпретации файла командным процессором вместо комбинации символов $1 осуществляется подстановка значения первого и единственного параметра andy.

Команда shift позволяет удалять первый параметр и сдвигает все остальные на места предыдущих. При использовании в командном файле комбинации символов $# вместо неё будет осуществлена подстановка числа параметров, указанных в командной строке при вызове данного командного файла на выполнение.

**Оператор цикла for**

В обобщённой форме оператор цикла for выглядит следующим образом: for имя [in список-значений] do список-команд done. [3]

# Выполнение лабораторной работы

1. Создадим новый файл lab12.sh. И текстовый файл if.txt с тексом стихотворения architecture И. Бродского (рис. 1)

* 
* Figure 1: Создание нового файла

1. Используя команды getopts grep, напишем командный файл, который анализирует командную строку с ключами (рис. 2):

– -iinputfile — прочитать данные из указанного файла;

– -ooutputfile — вывести данные в указанный файл;

– -pшаблон — указать шаблон для поиска;

– -C — различать большие и малые буквы;

– -n — выдавать номера строк.

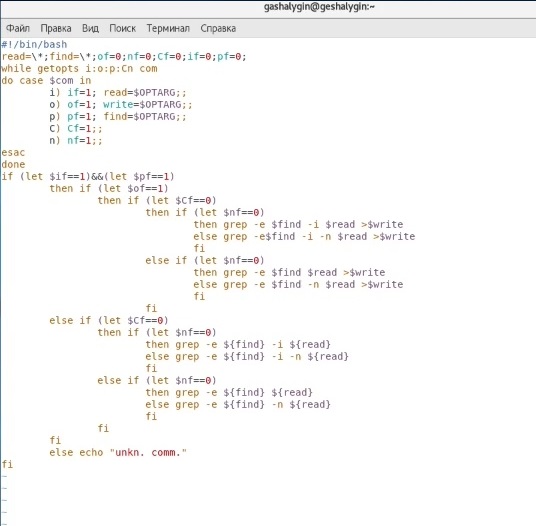
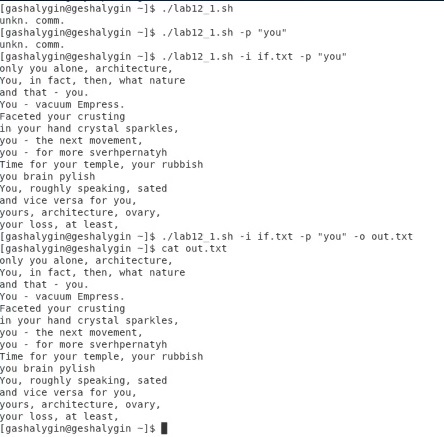
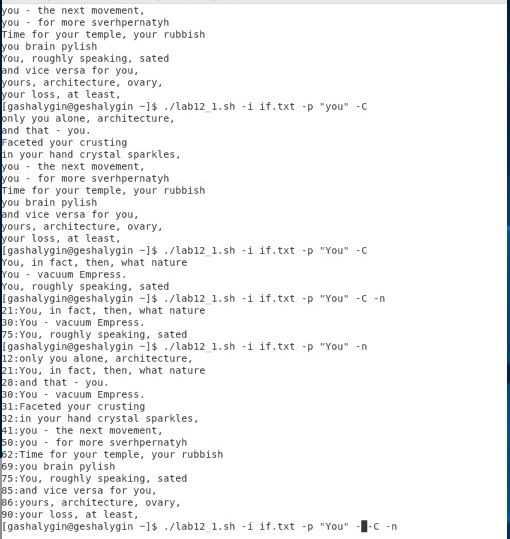
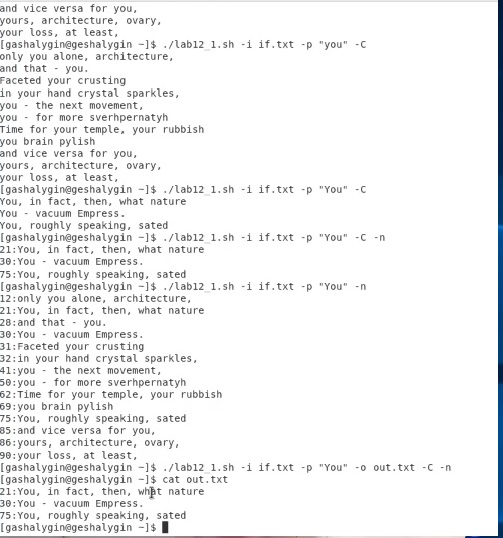
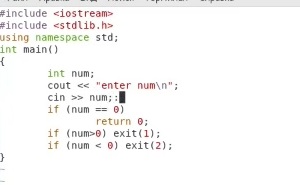


Figure 2: Текст 1 скрипта

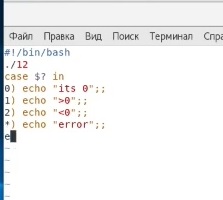
1. Результат работы скрипта, он выводит строки по запросу поиска и ключам. (рис. 3 4 5).

* 
* Figure 3: Результат поиска с ключами
* 
* Figure 4: Результат поиска с ключами
* 
* Figure 5: Результат поиска с ключами

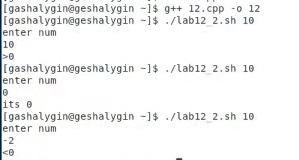
1. Напишем на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции exit(n), передавая информацию в о коде завершения в оболочку. (рис. 6)

* 
* Figure 6: Текст программы .срр

1. Командный файл вызывает эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено. (рис. 7)

* 
* Figure 7: Текст 3 скрипта

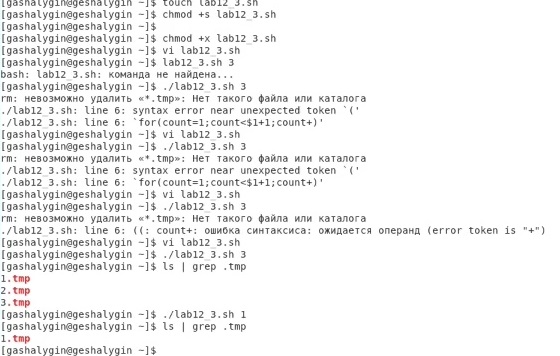
1. Результат работы (рис. 8).

* 
* Figure 8: Результат сравнения аргумента с 0

1. Напишем командный файл, создающий N файлов, пронумерованных последовательно от 1 до N (например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют). (рис. 9)

* 
* Figure 9: Текст 3 скрипта

1. Результат работы скрипта, сначала созданы 3 файла, затем - один, предыдущие удалены. (рис. 10)

* 
* Figure 10: Результат создания tmp файлов

1. Напишем командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории (рис. 11). Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). (рис. 12)



Figure 11: Текст 4 скрипта

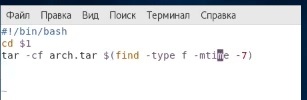


Figure 12: Текст модифицированного 4 скрипта

1. Результат работы скрипта, создан arch.tar. (рис. 13)



Figure 13: Результат создания архива

# Выводы

В процессе работы над лабораторной работы были изучены основы программирования в оболочке ОС UNIX, получен опыт написания более сложных командных файлов с использованием логических управляющих конструкций и циклов.

# Библиография

1. https://docs.altlinux.org/ru-RU/archive/2.3/html-single/junior/alt-docs-extras-linuxnovice/ch02s10.html
2. http://bourabai.kz/os/shells.htm
3. Д.С. Кулябов, А.В. Королькова / Администрирование локальных систем. Лабораторные работы. — М.: Российский университет дружбы народов, 2017. — 119 с.