Отчет по лабораторной работе 13

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Шалыгин Георгий Эдуардович, НФИбд-02-20

Содержание

# Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

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.

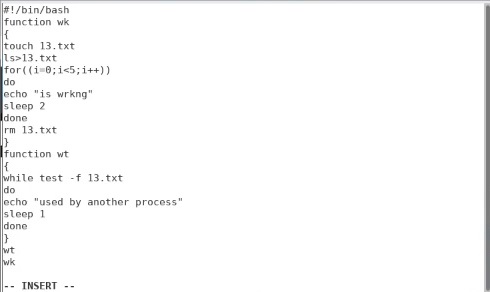
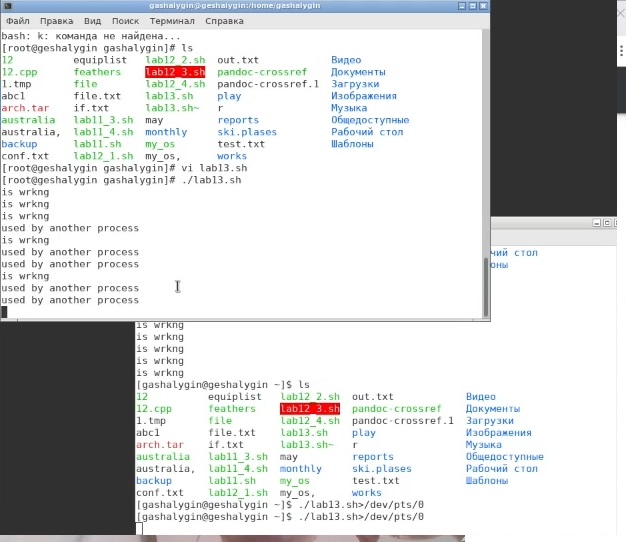
Команда shift позволяет удалять первый параметр и сдвигает все остальные на места предыдущих. При использовании в командном файле комбинации символов $# вместо неё будет осуществлена подстановка числа параметров, указанных в командной строке при вызове данного командного файла на выполнение.

**Оператор цикла for**

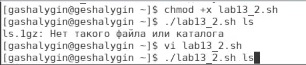
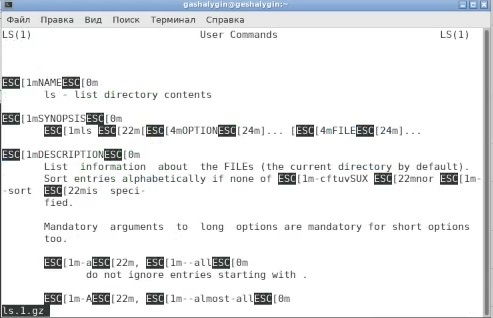
В обобщённой форме оператор цикла for выглядит следующим образом: for имя [in список-значений] do список-команд done. [3]

# Выполнение лабораторной работы

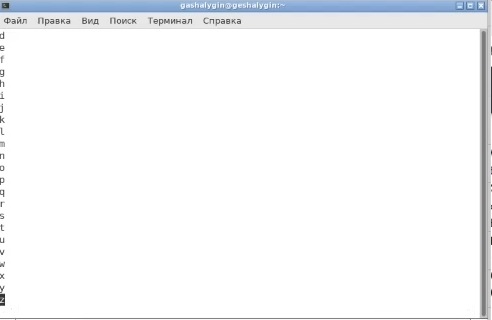
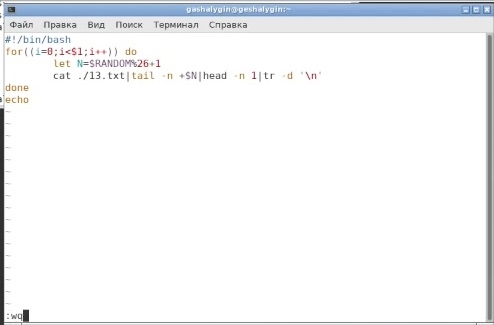
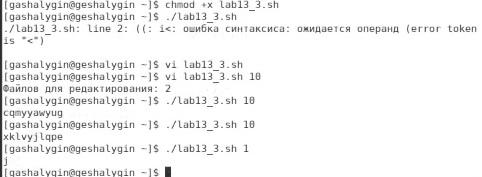
1. Напишем командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени t2<>t1, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (> /dev/tty#, где # — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме (рис. 1).

* Результат на (рис. 2).
* 
* Figure 1: Текст 1 скрипта
* 
* Figure 2: Результат работы 1 скрипта

1. Реализуем команду man с помощью командного файла. Изучим содержимое каталога /usr/share/man/man1. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой less сразу же просмотрев содержимое справки. Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге man1. (рис. 3 4 5).

* 
* Figure 3: Текст 2 скрипта
* 
* Figure 4: Результат работы 2 скрипта
* 
* Figure 5: Инфо о ls

1. Используя встроенную переменную $RANDOM, напишем командный файл, генерирующий случайную последовательность букв латинского алфавита. Т.к. $RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767, возьмем остаток по модулю 26. В файле 13.txt - 26 букв латинского алфавита. (рис. 6). Сам скрипт на рис 7

* 
* Figure 6: Текст 13.txt
* 
* Figure 7: Текст 3 скрипта
* Результат работы (рис. 8).
* 
* Figure 8: Результат генерации случайной последовательности букв

# 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.