Операционные системы

Лабораторная работа №10

Матюшкин Денис Владимирович (НПИбд-02-21)

Содержание

1 Цель работы

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

2 Ход работы

1. Напишем скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл архивируется архиваторов tar (рис. 2.1). Проверим работоспособность скрипта (рис. 2.2).

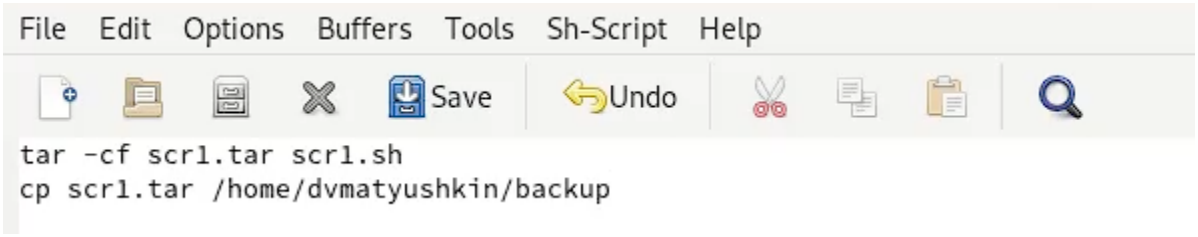

Рис. 2.1: Скрипт для копирования файла

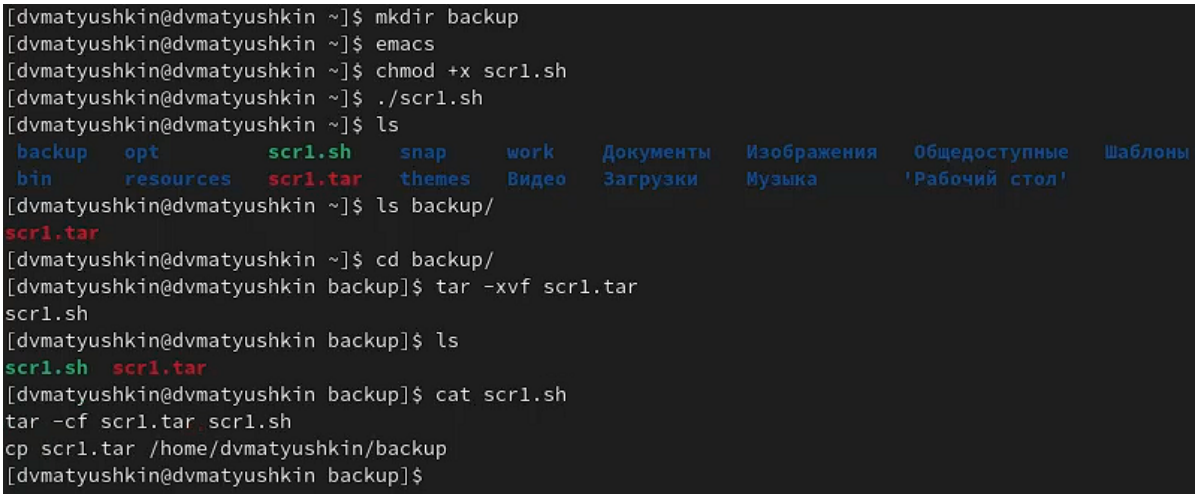

Рис. 2.2: Проверка скрипта

2. Напишем пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт последовательно печатает значения всех переданных аргументов (рис. 2.3). Проверим работоспособность скрипта (рис. 2.4).

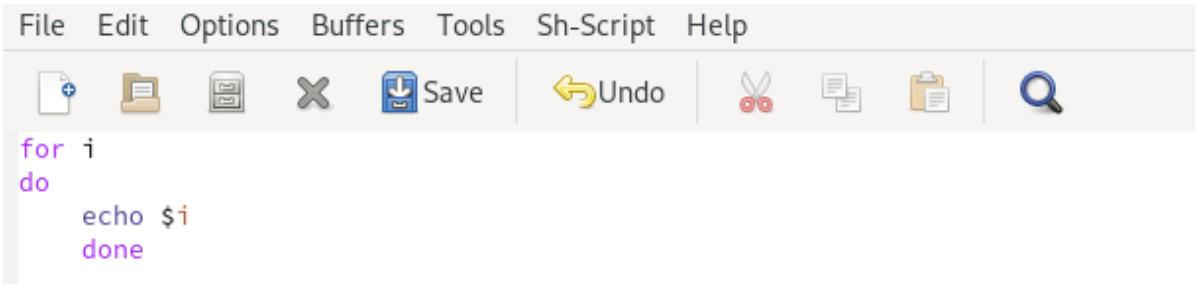

Рис. 2.3: Скрипт, который печатает аргументы

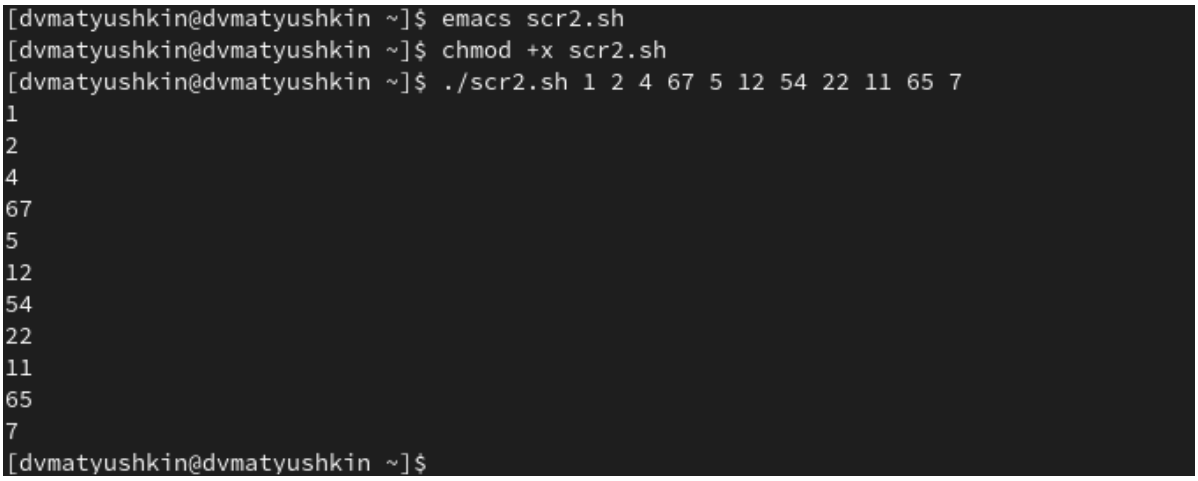

Рис. 2.4: Проверка скрипта

3. Напишем командный файл — аналог команды ls (без использования самой этой команды и команды dir). Выдает информацию о нужном каталоге и выводит информацию о возможностях доступа к файлам этого каталога (рис. 2.5). Проверим работоспособность скрипта (рис. 2.6).

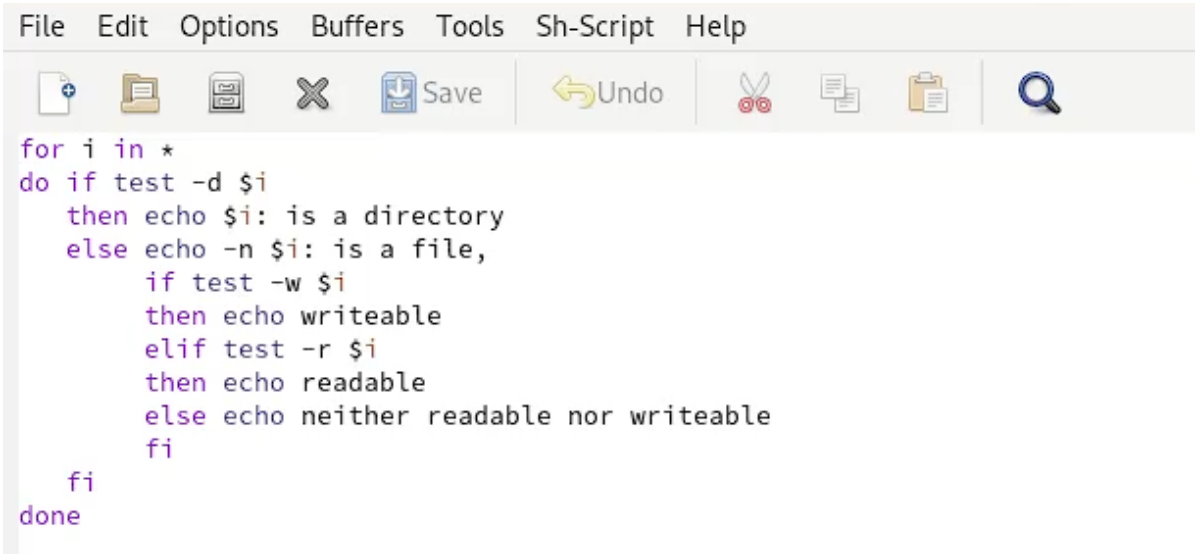

Рис. 2.5: Скрипт аналог ls с доп. выводом

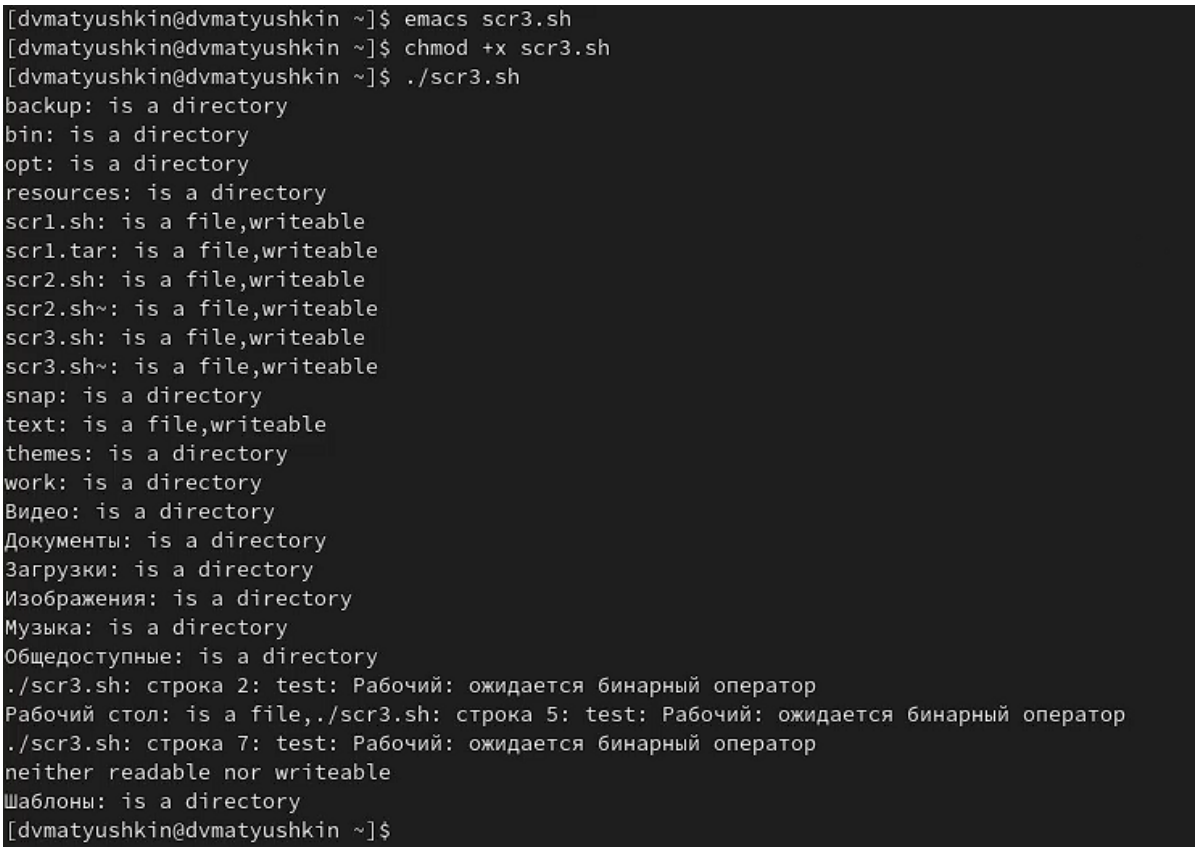

Рис. 2.6: Проверка скрипта

4. Напишем командный файл, который получает в качестве аргумента команд-

ной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаём в виде аргумента командной строки (рис. 2.7). Проверим работоспособность скрипта (рис. 2.8).

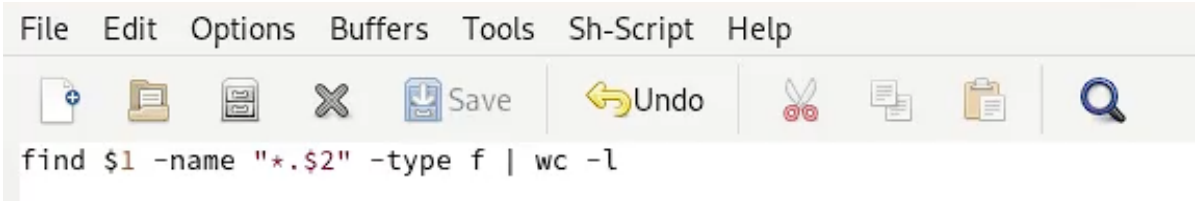

Рис. 2.7: Скрипт подсчета кол-ва файлов

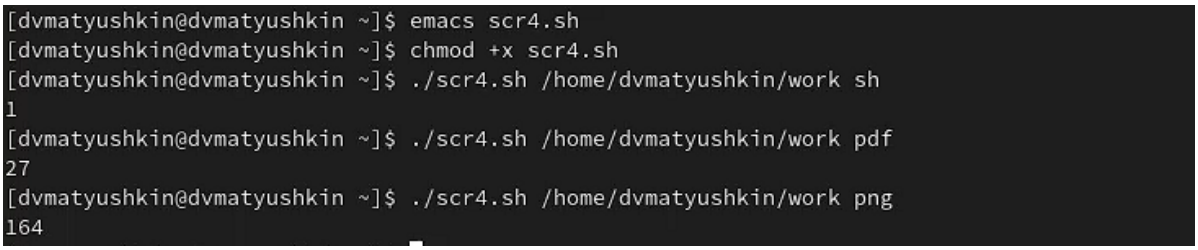

Рис. 2.8: Проверка скрипта

3 Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?

Программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.

Оболочка Борна - стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций.

C-оболочка - надстройка над оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд.

Оболочка Корна - напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна.

BASH - сокращение от Bourne Again Shell, в основе своей совмещает свойства оболочек C и Корна.

2. Что такое POSIX?

Набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

3. Как определяются переменные и массивы в языке программирования bash?

- Переменная/=значение.
- set -A (переменная), (список значений)

4. Каково назначение операторов let и read?

- let - берет два операнда и присваивает их переменной.

- read - чтение значения переменных со стандартного ввода.

5. Какие арифметические операции можно применять в языке программирования bash?

Операции логики, умножение, деление, сложение, вычитание.

6. Что означает операция (( ))?

Условия оболочки bash.

7. Какие стандартные имена переменных Вам известны?

PATH, IFS, MAIL, TERM, LOGNAME.

8. Что такое метасимволы?

Символы ’ < > * ? | ” &, являются метасимволами и имеют для командного процессора отличный от обычных символом смысл (они технически влияют на поведение программы).

9. Как экранировать метасимволы?

Экранирование может быть осуществлено с помощью предшествующего метасимволу символа , который, в свою очередь, является метасимволом. Для экранирования группы метасимволов нужно заключить её в одинарные кавычки. Строка, заключённая в двойные кавычки, экранирует все метасимволы, кроме $, ’, , “.

10. Как создавать и запускать командные файлы?

1. bash [аргументы]
2. chmod +x
3. ./командный_файл

11. Как определяются функции в языке программирования bash?

Ключевое слово function {тело функции}

12. Каким образом можно выяснить, является файл каталогом или обычным файлом?

– test -d file — истина, если file является каталогом, ложь - является файлом.

13. Каково назначение команд set, typeset и unset?

Оболочка bash позволяет работать с массивами. Для создания массива использу-

ется команда set с флагом -A typeset является встроенной инструкцией и предназначена для наложения ограничений на переменные.
С помощью команды unset можно изъять переменную из программы.

14. Как передаются параметры в командные файлы?

При вызове командного файла на выполнение параметры ему могут быть переданы точно таким же образом, как и выполняемой программе. С точки зрения командного файла эти параметры являются позиционными. Символ $ является метасимволом командного процессора. Он используется, в частности, для ссылки на параметры, точнее, для получения их значений в командном файле. В командный файл можно передать до девяти параметров. При использовании где-либо в командном файле комбинации символов $i, где 0 < i < 10, вместо неё будет осуществлена подстановка значения параметра с порядковым номером i, т.е. аргумента командного файла с порядковым номером i. Использование комбинации символов $0 приводит к подстановке вместо неё имени данного командного файла.

15. Назовите специальные переменные языка bash и их назначение.

- $* — отображается вся командная строка или параметры оболочки;
- $? — код завершения последней выполненной команды;
- $$ — уникальный идентификатор процесса, в рамках которого выполняется командный процессор;
- $! — номер процесса, в рамках которого выполняется последняя вызванная на выполнение в командном режиме команда;
- $- — значение флагов командного процессора;
- ${#} — возвращает целое число — количество слов, которые были результатом $;
- ${#name} — возвращает целое значение длины строки в переменной name;

- ${name[n]} — обращение к n-му элементу массива;
- ${name[*]} — перечисляет все элементы массива, разделённые пробелом;
- ${name[@]} — то же самое, но позволяет учитывать символы пробелы в самих переменных;
- ${name:-value} — если значение переменной name не определено, то оно будет заменено на указанное value;
- ${name:value} — проверяется факт существования переменной;
- ${name=value} — если name не определено, то ему присваивается значение value;
- ${name?value} — останавливает выполнение, если имя переменной не определено, и выводит value как сообщение об ошибке;
- ${name+value} — это выражение работает противоположно ${name-value}. Если переменная определена, то подставляется value;
- ${name#pattern} — представляет значение переменной name с удалённым самым коротким левым образцом (pattern);
- ${#name[*]} и ${#name[@]} — эти выражения возвращают количество элементов в массиве name.

4 Вывод

- В ходе этой лабораторной работы мы изучили основы программирования в оболочке ОС UNIX/Linux. Научились писать небольшие командные файлы.