Отчёт по лабораторной работе №10

Программирование в командном процессоре ОС UNIX. Командные файлы.

Смирнов-Мальцев Егор Дмитриевич

# Цель работы

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

# Задание

1. Написать 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 должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку.
2. Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.
3. Написать командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.
4. Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки.

# 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 С-подобный 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/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна.

# Выполнение лабораторной работы

1. Написал 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 архиватором tar (рис. 1).

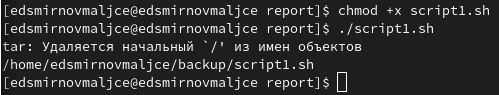


рис. 1: выполнение первого скрипта.

1. Написал пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Скрипт последовательно распечатывает значения всех переданных аргументов (рис. 2).

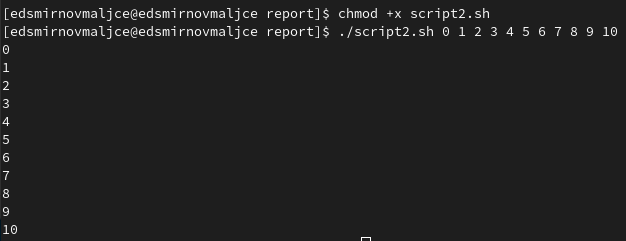


рис. 2: выполнение второго скрипта.

1. Написал командный файл — аналог команды ls (без использования самой этой команды и команды dir). Он выдает информацию о нужном каталоге и выводит информацию о возможностях доступа к файлам этого каталога (рис. 3).

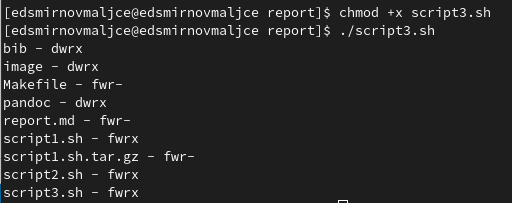


рис. 3: выполнение третьего скрипта.

1. Написал командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки (рис. 4).

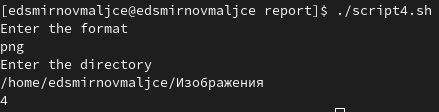


рис. 4: выполнение четвертого скрипта.

# Ответы на контрольные вопросы

1. Командная оболочка или командный интерпретатор - это приложение предоставляющее пользователю интерфейс командной строки в которой тот либо вводит команды по отдельности, либо запускает скрипты состоящие из списка команд.

Список существующих оболочек велик. Условно можно разделить их на три группы:

* Клоны Bourne shell (bash, zsh)
* C shell (csh, tcsh)
* Базирующиеся на популярных языках программирования(psh, ipython, scsh)
* Экзотические, специфические и все остальные

Они отличаются синтаксисом команд.

1. POSIX (англ. Portable Operating System Interface — переносимый интерфейс операционных систем) — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов.
2. Имена переменных могут быть выбраны пользователем. Например, переменную можно объявить так:

mark=value

Массив объявляется с помощью команды set с флагом -A:

set -A massive 1 2 3 4

1. let показывает, что следующие аргументы являются выражением, подлежащим вычислению. read говорит о том. что в последующую переменную надо записать значение, прочитанное с командной строки.
2. Можно складывать, вычитать, перемножать, целочисленно делить, считать остаток от деления, совершать побитовые XOR, OR, AND, сдвигать не нсколько бит влево и вправо. Также есть унарный минус.
3. Операция (( )) показывает, что внутри расположено условие.
4. Есть следующие стандартные имена: Var1, PATH, trash, mon, day, PS1, PS2, -HOME, -IFS, -MAIL, -TERM, -LOGNAME.
5. Метасимволы - символы, которые несут отдельный смысл для компилятора.
6. Метасимволы экранируются заключением их в одинарные скобки.
7. Командные файлы создаются в тестовыхредакторах и сохранются в расширении .sh. Далее его можно выполнить по команде bash.

bash script.sh [args]

1. Группу команд можно объединить в функцию с помощью слова function:

function func\_name{  
 body  
}

1. Определить является ли объект каталогом с помощью команды test -d. Если это директория, команда выведет 1 иначе 0.
2. set -A - определяет массив. typeset - определяет тип последующей переменной. unset - удаляет переменную.
3. Параметры передаются при запуске скрипта. Обратится к параметру можно с помощью $i, где i - номер параметра.
   * $\* — 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.

# Выводы

* Я научился писать простые программы на языке bash.