Отчёт по лабораторной работе № 11

Операционные системы

Ильина Любовь Александровна

Содержание

# Цель работы

Изучение основ программирования в оболочке ОС UNIX/Linux. Написание небольших командных файлов.

# Выполнение лабораторной 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 должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. (рис. [1](#fig:001) - [2](#fig:002))

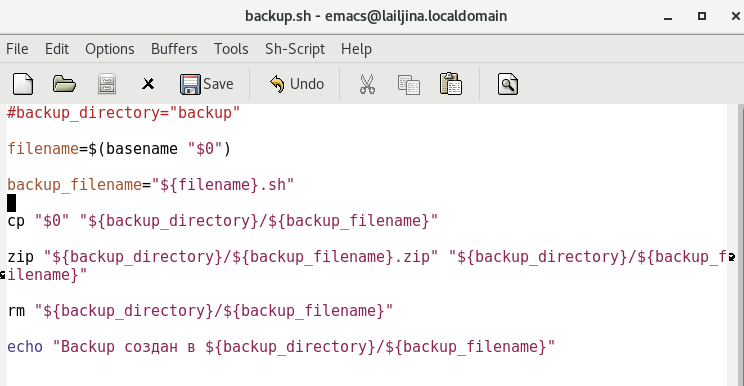


Figure 1: Скрипт командного файла, создающего свой backup

Figure 2: Выполнение командного файла, создающего свой backup

Figure 2: Выполнение командного файла, создающего свой backup

1. 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. (рис. [3](#fig:003) - [4](#fig:004))

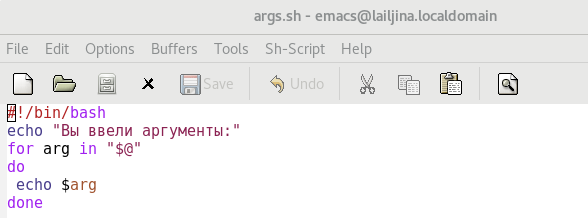


Figure 3: Скрипт командного файла, обрабатывающего любое произвольное число аргументов

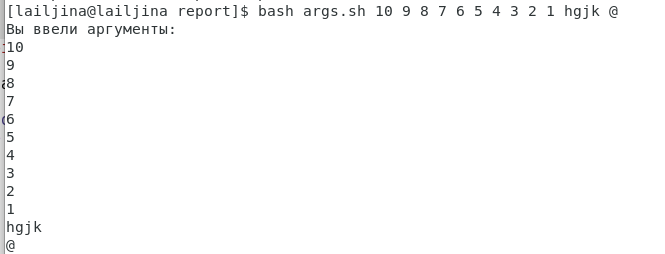


Figure 4: Выполнение командного файла, обрабатывающего любое произвольное число аргументов

1. 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 (рис. [5](#fig:005) - [6](#fig:006))



Figure 5: Скрипт командного файла, аналога команды ls

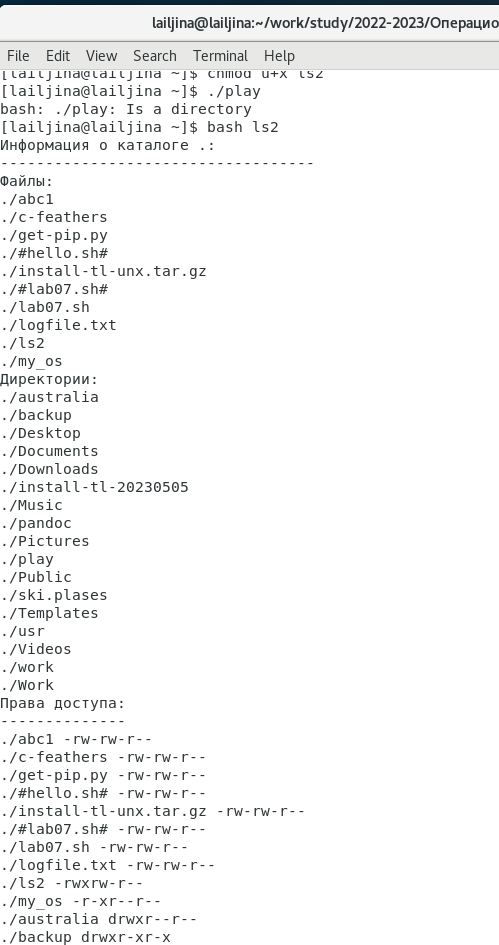


Figure 6: Выполнение командного файла, аналога команды ls

1. 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 (рис. [7](#fig:007) - [8](#fig:008))

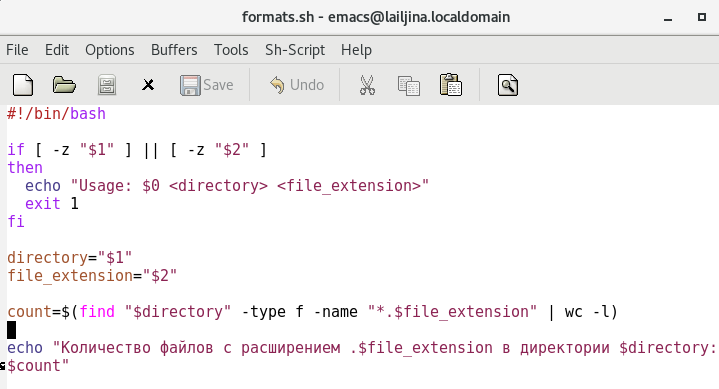


Figure 7: Скрипт командного файла, который вычисляет количество файлов указанного формата в указанной директории

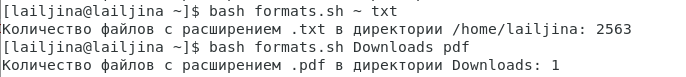


Figure 8: Выполнение командного файла, который вычисляет количество файлов указанного формата в указанной директории

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

1. Что такое POSIX? POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ
2. Как определяются переменные и массивы в языке программирования bash? Командный процессор bash обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Присвоение с помощью =. Значение, присвоенное некоторой переменной, может быть впоследствии использовано по символу $. Например, для переменной mark: mv afile ${mark}. Оболочка bash позволяет работать с массивами. Для создания массива используется команда set с флагом -A. За флагом следует имя переменной, а затем список значений, разделённых пробелами. Например, set -A states Delaware Michigan “New Jersey”
3. Каково назначение операторов let и read? Команда let является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Простейшее выражение — это единичный терм (term), обычно целочисленный. Команда let также расширяет другие выражения let, если они заключены в двойные круглые скобки. Таким способом вы можете создавать довольно сложные выражения. Команда let не ограничена простыми арифметическими выражениями. Команда read позволяет читать значения переменных со стандартного ввода: echo “Please enter Month and Day of Birth ?” read mon day trash В переменные mon и day будут считаны соответствующие значения, введённые с клавиатуры, а переменная trash нужна для того, чтобы отобрать всю избыточно введённую информацию и игнорировать её.
4. Какие арифметические операции можно применять в языке программирования bash? (рис. [9](#fig:009))

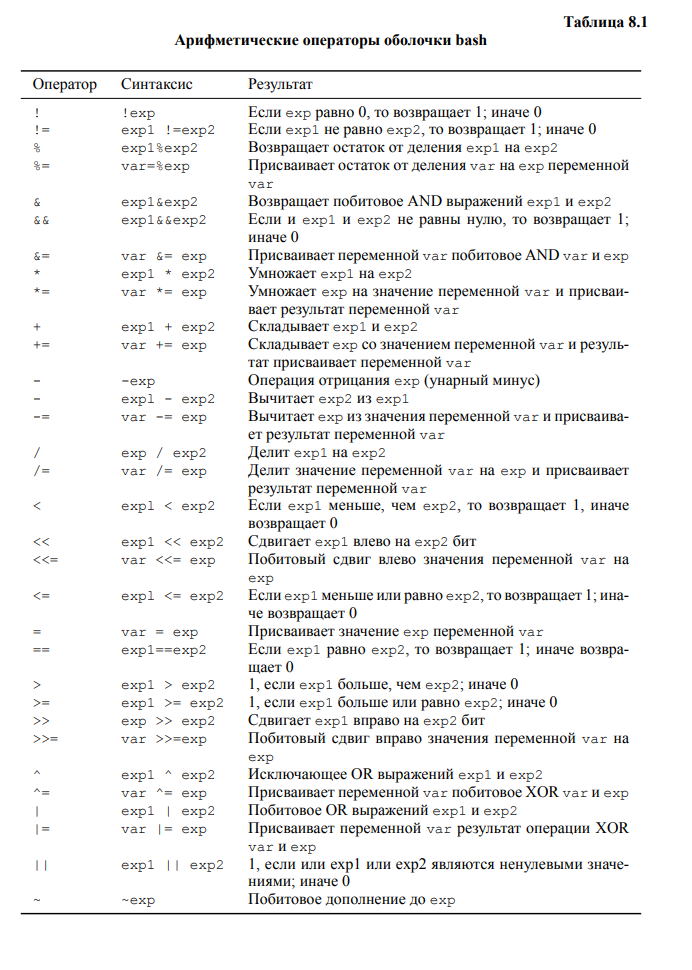


Figure 9: Арифметические операции bash

1. Что означает операция (())? условия оболочи let для удобства вкладывают в (( ))
2. Какие стандартные имена переменных Вам известны? PATH - стандартная (по умолчанию) последовательность поиска файла: текущий каталог, каталог /bin, каталог /usr/bin. Переменной можно присвоить иное значение. – HOME — имя домашнего каталога пользователя. Если команда cd вводится без аргументов, то происходит переход в каталог, указанный в этой переменной. – IFS — последовательность символов, являющихся разделителями в командной строке, например, пробел, табуляция и перевод строки (new line). – MAIL — командный процессор каждый раз перед выводом на экран промптера проверяет содержимое файла, имя которого указано в этой переменной, и если содержимое этого файла изменилось с момента последнего ввода из него, то перед тем как вывести на терминал промптер, командный процессор выводит на терминал сообщение You have mail (у Вас есть почта). – TERM — тип используемого терминала. – LOGNAME — содержит регистрационное имя пользователя, которое устанавливается автоматически при входе в систему.
3. Что такое метасимволы? Такие символы, как ’ < > \* ? |  " &, являются метасимволами и имеют для командного процессора специальный смысл. Снятие специального смысла с метасимвола называется экранированием метасимвола.
4. 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 $, ’ , , ". Например, – echo \* выведет на экран символ *, – echo ab’*|*’cd выведет на экран строку ab*|\*cd.
5. Как создавать и запускать командные файлы? Создание файла c помощью редактора vim: vim .sh. Также можно создать командой touch и редактировать в терминале: cat > .sh. Сохранить изменения по Исполнение файла: chmod u+x .sh Вы можете запустить скрипт любым из указанных способов: sh .sh bash .sh ./.sh
6. Как определяются функции в языке программирования bash? Группу команд можно объединить в функцию. Для этого существует ключевое слово function, после которого следует имя функции и список команд, заключённых в фигурные скобки.
7. Каким образом можно выяснить, является файл каталогом или обычным файлом? for A in \* do if test -d $A then echo $A: is a directory else echo -n $A: is a file
8. Каково назначение команд set, typeset и unset? Команда set — это встроенная команда оболочки, которая позволяет отображать или устанавливать переменные оболочки и среды. Команда typeset имеет четыре опции для работы с функциями: – -f — перечисляет определённые на текущий момент функции; – -ft — при последующем вызове функции инициирует её трассировку; – -fx — экспортирует все перечисленные функции в любые дочерние программы оболочек; – -fu — обозначает указанные функции как автоматически загружаемые. Автоматически загружаемые функции хранятся в командных файлах, а при их вызове оболочка просматривает переменную FPATH, отыскивая файл с одноимёнными именами функций, загружает его и вызывает эти функции Удалить функцию можно с помощью команды unset c флагом -f.
9. Как передаются параметры в командные файлы? Символ $ используется для ссылки на параметры, точнее, для получения их значений в командном файле.
10. Назовите специальные переменные языка bash и их назначение $# - осуществляет подстановку числа параметров, указанных в командной строке при вызове данного командного файла на выполнение. – $\* — 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

Изучила основы программирования в оболочке ОС UNIX/Linux. Научилась писать небольшие командные файлы.