Отчёт о выполнении лабораторной работы №10

Российский Университет Дружбы Народов
Факультет Физико-Математических и Естественных Наук

Дисциплина: *Операционные системы*

Работу выполняла: *Арежина Адриана*

№ ст. билета: *1032201674*

Группа: *НКНбд-01-20*

Москва. 2021г.

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.

Выполнение работы

1. Изучила справку *tar*. (см. рисунок ниже tar)

Написала скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию *backup* в домашнем каталоге. (см. рисунки ниже файл 1, результат 1)

2. Написала пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Скрипт может последовательно распечатывать значения всех переданных аргументов. (см. рисунки файл 2, результат 2)

3. Написала командный файл — аналог команды *ls* (без использования самой этой команды и команды *dir*), который выдаёт информацию о нужном каталоге и выводит информацию о возможностях доступа к файлам этого каталога. (см. рисунки файл 3, результат 3)

4. Написала 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. (см. рисунки файл 4, результат 4)

Контрольные вопросы

1. Командные процессоры или оболочки - это программы, позволяющие пользователю взаимодействовать с компьютером. Их можно рассматривать как настоящие интерпретируемые языки, которые воспринимают команды пользователя и обрабатывают их. Поэтому командные процессоры также называют интерпретаторами команд. На языках оболочек можно писать программы и выполнять их подобно любым другим программам.

Примеры: - Оболочка Борна (Bourne) - первоначальная командная оболочка UNIX: базовый, но полный набор функций; - C - оболочка - добавка университета Беркли к коллекции оболочек: она надстраивается над оболочкой Борна, используя C-подобный синтаксис команд, и сохраняет историю выполненных команд; - Оболочка Корна - напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; - BASH - сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).

2. POSIX (Portable Operating System Interface for Computer Environments)- интерфейс переносимой операционной системы для компьютерных сред. Представляет собой набор стандартов, подготовленных институтом инженеров по электронике и радиотехники (IEEE), который определяет различные аспекты построения операционной системы. POSIX включает такие темы, как программный интерфейс, безопасность, работа с сетями и графический интерфейс. POSIX-совместимые оболочки являются будущим поколением оболочек UNIX и других ОС. Windows NT рекламируется как система, удовлетворяющая POSIX-стандартам. POSIX-совместимые оболочки разработаны на базе оболочки Корна; фонд бесплатного программного обеспечения (Free Software Foundation) работает над тем, чтобы и оболочку BASH сделать POSIX-совместимой.

3. 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. Значение, присвоенное некоторой переменной, может быть впоследствии использовано. Для этого в соответствующем месте командной строки должно быть употреблено имя этой переменной, которому предшествует метасимвол $. Например, команда mv afile $mark переместит файл afile из текущего каталога в каталог с абсолютным полным именем /usr/andy/bin. 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} например, использование команд b=/tmp/andy-ls -l myfile > ${b}ls приведет к переназначению стандартного вывода команды ls с терминала на файл /tmp/andy-ls , а использование команды ls -l>$bls приведет к подстановке в командную строку значения переменной bls. Если переменной bls не было предварительно присвоено никакого значения, то ее значением является символ пробел. Оболочка bash позволяет создание массивов. Для создания массива используется команда set с флагом -A. За флагом следует имя переменной, а затем список значений, разделенных пробелом. Например,set -A states Delaware Michigan "New Jersey" Далее можно сделать добавление в массив, например, states[49]=Alaska. Индексация массивов начинается с нулевого элемента.

4. Команда let является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Простейшее выражение - это единичный терм (term), обычно целочисленный. Целые числа можно записывать как последовательность цифр или в любом базовом формате. Этот формат — radix#number, где radix (основание системы счисления) - любое число не более 26 Для большинства команд основания систем счисления это - 2 (двоичная), 8 (восьме- ричная) и 16 (шестнадцатеричная). Простейшими математическими выражениями являются сложение (+), вычитание (-), умножение (*), целочисленное деление (/) и целочисленный остаток (%). Команда let берет два операнда и присваивает их переменной. Команда read читает одну строку из стандартного входного потока и записывает ее содержимое в указанные переменные. Если задана единственная переменная, в нее записывается вся строка. В результате ввода команды read без параметров строка помешается в переменную среды $reply. При указании нескольких переменных в первую из них записывается первое слово строки, во вторую — второе слово и т. д. Последней переменной присваивается остаток строки.

5. *+* сложение

- вычитание

*** умножение

/ деление

_**_ возведение в степень

% остаток от деления

6. Внутри (( )) вычисляются арифметические выражения и возвращается их результат. Например, в простейшем случае, конструкция a=$((5 + 3)) присвоит переменной «a» значение выражения «5 + 3», или 8 Кроме того, двойные круглые скобки позволяют работать с переменными в стиле языка C.

7. $BASH

В переменной $BASH содержится полный путь до исполняемого файла командной оболочки Bash.

$BASH_VERSION

В переменную $BASH_VERSION записывается версия Bash.

$CDPATH

Переменная, которая хранит пути поиска каталога. (используется при вводе команды cd имя_каталога без слэша)

$CLASSPATH

содержит список каталогов для поиска файлов классов Java и архивов Java.

$HOME

домашний каталог текущего пользователя.

$HOSTNAME

В переменной $HOSTNAME хранится имя компьютера.

$HISTSIZE

количество событий, хранимых в истории за 1 сеанс

$HISTFILE

Расположение файла истории событий

$HISTFILESIZE

количество событий, хранимых в истории между сеансами

$IFS

переменная хранит символы, являющиеся разделителями команд и параметров. (по умолчанию - пробел, табуляция и новая строка)

$LANG

текущая установка локализации, которая позволяет настроить командную оболочку для использования в различных странах и на различных языках.

$MAIL

Место, где храниться почта

$OSTYPE

В переменной $OSTYPE содержится описание операционной системы.

$PATH

список каталогов для поиска команд и приложений, когда полный путь к файлу не задан.

$PS1

PS1 используется как основная строка приглашения. (то самое [root@proxy ~]#)

$PS2

PS2 используется как вторичная строка приглашения.

$PROMPT_COMMAND

Эта команда должна быть выполнена до отображения строки приглашения Bash.

$PWD

полный путь к текущему рабочему каталогу.

$SHELL

полный путь к текущей командной оболочке.

$USER

В переменной $USER содержится имя текущего пользователя.

8. Такие символы, как ’ < > * ? | \ " & . Они имеют для командного процессора специальный смысл.
9. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа\, который, в свою очередь, является метасимволом. Для экранирования группы метасимволов, ее нужно заключить в одинарные кавычки. Строка, заключенная в двойные кавычки, экранирует все метасимволы, кроме $, ’ , \, ". Например,–echo *выведет на экран симво*л, –echo ab*|’cd выдаст строку ab*|*cd.
10. Для создания командного файла: Запустить текстовый редактор. Последовательно записать команды, располагая каждую команду на отдельной строке. Сохранить этот файл, сделать его исполняемым, применив команду: chmod +x имя_файла. Запустить этот файл можно или используя команду sh ~/cmd.
11. Группу команд можно объединить в функцию. Для этого существует ключевое слово function, после которого следует имя функции и список команд, заключенных в фигурные скобки. Удалить функцию можно с помощью команды unset с флагом-f. Команда typeset имеет четыре опции для работы с функциями: -f — перечисляет определенные на текущий момент функции; –-ft— при последующем вызове функции инициирует ее 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 одноименными именами функций, загружает его и вызывает эти функции.
12. ls -lrt Если есть d, то является файл каталогом.
13. Команда set изменяет значения внутренних переменных сценария.

Команда typeset задаёт и/или накладывает ограничения на переменные.

Команда unset удаляет переменную, фактически -- устанавливает ее значение в null.

14. В командный файл можно передать до девяти параметров. При использовании где-либо в команд- ном файле комбинации символов $i, где 0 < < 10, вместо нее 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 нее имени данного командного файла. Рассмотрим это на примере. Пусть к командному файлу where имеется доступ по выполнению и этот командный файл содержит следующий конвейер: who | grep $1 Если Вы введете с терминала команду: where andy, то в случае, если пользователь, зарегистрированный в ОС UNIX под именем andy, в данный момент работает в ОС UNIX, на терминал будет выведена строка, содержащая номер терминала, используемого указанным пользователем. Если же в данный момент этот пользователь не работает в ОС UNIX, то на терминал не будет выведено ничего. Команда grep производит контекстный поиск в тексте, поступающем со стандартного ввода, для нахождения в этом тексте строк, содержащих последовательности символов, переданные ей в качестве аргументов, и выводит результаты своей работы на стандартный вывод. В этом примере команда grep используется как фильтр, обеспечивающий ввод со стандартного ввода и вывод всех строк, содержащих последовательность символов andy, на стандартный вывод. В ходе интерпретации этого файла командным процессором вместо комбинации символов $1 осуществляется подстановка значения первого и единственного параметра andy. Если предположить, что пользователь, зарегистрированный в ОС UNIX под именем andy, в данный момент работает в ОС UNIX, то на терминале Вы увидите примерно следующее: $ where andy andy ttyG Jan 14 09:12 $ Определим функцию, которая изменяет каталог и печатает список файлов: $ function clist { > cd $1 > ls > }. Теперь при вызове команды clist каталог будет изменен каталог и выведено его содержимое.
15. - $* — 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-ному элементу массива; - ${name[*]} — *перечисляет все элементы массива, разделенные про- белом; - ${name[@]} — то же самое, но позволяет учитывать символы про- белы 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 удаленным самым коротким левым образцом (pattern); - ${#name[*]}* и ${#name[@]} — эти выражения возвращают количество элементов в массиве name. - $# вместо нее будет осуществлена подстановка числа параметров, указанных в командной строке при вызове данного командного файла на выполнение.

Вывод

Я изучила основы программирования в оболочке ОС UNIX/Linux и научилась писать небольшие командные файлы.