**Лабораторная работа № 4 - Командные файлы. Переменные окружения**

**Цель работы**

Знакомство с важным атрибутом любой операционной системы – переменными среды (или переменными окружения) и с возможностями их использования в Linux. Освоение языка для составления командных сценариев и написание набора полезных для системного администрирования скриптов.

**Пункт 1**

Создайте несколько символьных переменных среды (переменных окружения). Составьте командный файл (сценарий bash), выводящий на консоль значения этих переменных. Выполните операцию конкатенации (склеивания) значений переменных и выведите полученный результат на консоль. Выделите из конкатенированной переменной среды подстроку и выведите ее на консоль. Замените выделенную подстроку на какое-либо другое значение и выведите измененное значение переменной среды на консоль.

1. Создадим несколько символьных переменных среды по средствам использования команды export, которая экспортирует переменные и функции дочерним процессам. Затем найдем обе переменные среды и добавим их в одну строку(конкатенация). Команда env отобразит "глобальные" переменные доступные всем. Далее выделим из строки подстроку, которая содержит значение первой символьной переменной среды, а затем выведем изменённое значение первой символьной переменной.

|  |
| --- |
| Листинг 4.1 Выведение переменных среды (file1.sh) |
| #!/bin/bash  export VAR1="sneg"  export VAR2="padaet"  env | grep VAR\*  NEW\_VAR="$VAR1 $VAR2"  echo $NEW\_VAR  echo ${NEW\_VAR:0:${#VAR1}}  echo ${NEW\_VAR/$VAR1/"kaplya"} |

1. Запустим скрипт:

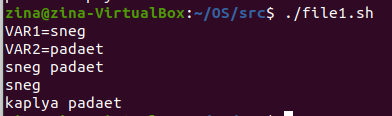


Рис. 4-1 Результат выполнения

В результате выполнения скрипта был получены ожидаемые результаты.

**Пункт 2**

Создайте несколько переменных среды в интерпретации, как числовые переменные. В новом командном файле выполните с этими числовыми переменными все допустимые арифметические операции, выводя на консоль результаты операций и соответствующие комментарии.

1. Создадим скрипт, который будет выполнять с переменными среды все возможные арифметические операции, а именно сложение, вычитание, умножение, целая часть от деления, остаток от деления и возведение в степень.

|  |
| --- |
| Листинг 4.2 Работа с арифметическими операциями (file2.sh) |
| #!/bin/bash  export VAR1=8  export VAR2=2  echo -n "$VAR1 + $VAR2 = "  let val1=VAR1+VAR2  echo "$val1 - addition of numbers"  echo ""  echo -n "$VAR1 - $VAR2 = "  let val2=VAR1-VAR2  echo "$val2 - subtract of numbers"  echo ""  echo -n "$VAR1 \* $VAR2 = "  let val3=VAR1\*VAR2  echo "$val3 - multiplication of numbers"  echo ""  echo -n "$VAR1 / $VAR2 = "  let val4=VAR1/VAR2  echo "$val4 - division of numbers"  echo ""  echo -n "$VAR1 % $VAR2 = "  let val5=VAR1%VAR2  echo "$val5 - remainder of division of numbers"  echo ""  echo -n "$VAR1 \*\* $VAR2 = "  let val6=VAR1\*\*VAR2  echo "$val6 - raising $VAR1 to the power of $VAR2"  echo "" |

В языке bash у переменных нет типов данных, значение любой переменной — это строка. Команда let производит математические операции над переменными.

1. Запустим скрипт:

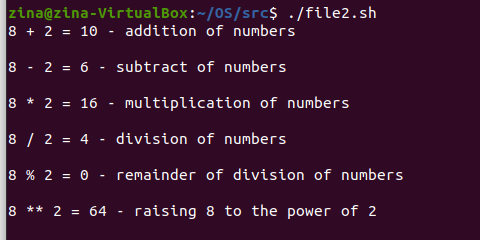


Рис. 4-2 Результаты выполнения

Полученные результаты совпадают с ожидаемыми.

**Пункт 3**

Создайте командный файл (основной), выдающий при старте сообщение и затем вызывающий другой командный файл (его имя задается при старте основного файла в качестве параметра командной строки), который выдает свое сообщение и приостанавливается до нажатия любой клавиши. При возврате управления в вызывающий (основной) файл из него должно выдаваться еще одно сообщение, подтверждающее возврат.

1. Создадим командных файла, один из которых будет запускать другой. Для этого воспользуемся командой sh, которая запускает новый экземпляр оболочки.

|  |
| --- |
| Листинг 4.3.1 Основной командный файл (file3\_1.sh) |
| #!/bin/bash    echo "This script is about to run another script"  sh ./$1  echo "This script has already ran another script" |

|  |
| --- |
| Листинг 4.3.2 Второй командный файл (file3\_2.sh) |
| #!/bin/bash    echo "Input something"  read a |

1. Запустим скрипт:

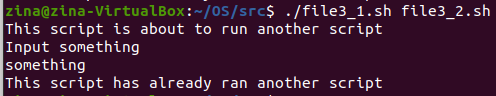


Рис. 4-3 Результат выполнения

Сначала в консоли выводится сообщение о том, что нас основной скрипт запускает другой скрипт. Второй скрипт ожидает, пока пользователь не введет что-нибудь в консоль. После того, как пользователь произвёл ввод, выводится сообщение о том, что основной скрипт уже запустил наш второй скрипт. Результаты совпадают с требуемыми.

**Пункт 4**

Составьте командный файл, выводящий на экран различия содержимого двух каталогов, имена которых передаются в качестве параметров. Отличия искать в именах файлов, их размерах и атрибутах.

1. Составим командный файл.

|  |
| --- |
| Листинг 4.4 Вывод различий между каталогами |
| #!/bin/bash    (cd $1 && ls -Ggh1 | awk '{print $1, $3, $7}') > "tmp1.out"  (cd $2 && ls -Ggh1 | awk '{print $1, $3, $7}') > "tmp2.out"  while IFS= read -r line2  do  while IFS= read -r line1  do  foundEq=false  if [[ "$line1" == "$line2" ]]; then  foundEq=true  break  fi  done < "./tmp1.out"  if [[ "$foundEq" != "true" ]]; then  echo "no pair found for :"  echo "$line2 from dir2"  echo ""  fi  done < "./tmp2.out"    while IFS= read -r line1  do  while IFS= read -r line2  do  foundEq=false  if [[ "$line1" == "$line2" ]]; then  foundEq=true  break  fi  done < "./tmp2.out"  if [[ "$foundEq" != "true" ]]; then  echo "no pair found for :"  echo "$line1 from dir1"  echo ""  fi  done < "./tmp1.out"    rm "tmp1.out"  rm "tmp2.out" |

Первоначально, мы переходим в директории, которые собираемся сравнивать, и получаем информацию о файлах, которые в них содержатся. Команда awk построчно читает файлы и добавляет во временные файлы первый, третий и седьмой атрибут.

Затем мы начинаем перебирать строки во временных файлах пытаясь найти совпадающие. Если таковые не были найдены, то выводится сообщение о том, что соответствующему файлу из первой директории не был найден аналогичный из второй директории. Данную операцию производим два раза, сначала сравнивая первую директорию со второй, а затем вторую с первой. По окончании выполнения удаляем временные файлы.

1. Произведем сравнение двух директорий.

|  |  |
| --- | --- |
|  |  |
| Рис. 4-4 Первая директория | Рис. 4-5 Вторая директория |

Файлы ex01 будут иметь разный размер, но одинаковое имя. Файлы ex02 будут иметь разные права доступа (в dir2 для файла ex02 запретим запись всем, кроме владельца, с помощью команды «chmod 755 ex02»).

Файл \*.docx будет находиться в обеих директориях, а файл \*.xlsx будет находиться только в одной из двух директорий.

1. Запустим командный файл

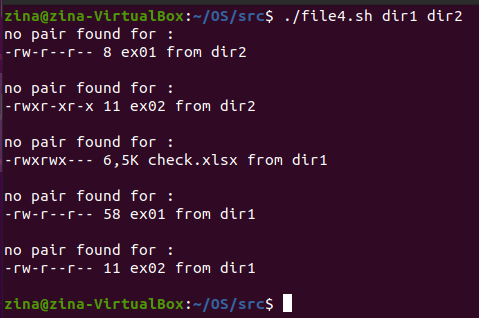


Рис. 4-6 Результаты выполнения

Из результатов мы видим, что пара была найдена только для файла \*.dоcx, который содержался в обеих директориях и был идентичен файлу из другой директории.

**Пункт 5**

Разработайте командный файл сценария для поиска текстовых файлов, содержащих заданную последовательность символов. Эта последовательность передается при запуске в качестве первого параметра командной строки. В качестве второго параметра передается имя файла результатов, который должен быть создан в сценарии для записи в него имен найденных текстовых файлов и номеров их строк, в которых содержится заданная последовательность символов.

1. Сначала находим все файлы в текущей папке, которые содержат заданную последовательность символом и заносим их во временный файл. Затем в файл, название которого соответствует второму параметру, добавляем информацию о пути к файлу, номере строки, в которой была найдена подстрока, и саму подстроку.

|  |
| --- |
| Листинг 4.5 Поиск файлов, содержащих заданную последовательность |
| grep -Ril $1 $(pwd) > "./tmp.out"    while IFS= read -r line  do  echo "$line $(cat -b $line | grep $1)" >> "./$2"  done < "./tmp.out"  rm "./tmp.out" |

1. Запустим командный файл:



Рис. 4-7 Запуск скрипта

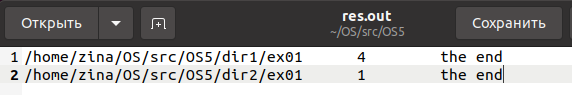


Рис. 4-8 Результат выполнения скрипта

В результате выполнения мы видим, что было найдено два файла, которые имеют заданную нами подстроку. Давайте посмотрим, что находилось внутри этих файлов.

1. Создадим два файла в разных дерикториях:

|  |  |
| --- | --- |
|  |  |
| Рис. 4-9 Содержимое файла в dir1 | Рис. 4-10 Содержимое файла в dir2 |

Как мы видим, файл в директории dir1 в четвёртой строке имеет введенную нами подстроку, а файл в директории dir2 имеет в первой строке введенную нами подстроку.

**Пункт 6**

Дубликат задания из Пункта 5.

**Пункт 7**

Создайте командный файл, который синхронизирует содержимое заданного каталога с эталонным. После запуска и отработки командного файла в заданном каталоге должен оказаться тот же набор файлов, что и в эталонном (если файла нет – он копируется из эталонного каталога, если найдется файл, которого нет в эталонном, – удаляется). Если файл с некоторым именем есть и в заданном, и в эталонном каталогах, то он перезаписывается только в том случае, если в эталонном имеется более новая версия файла. Имена обоих каталогов должны при запуске передаваться командному файлу параметрами командной строки.

1. Первоначально, мы переходим в директории, которые собираемся сравнивать, и получаем информацию о файлах, которые в них содержатся. Команда awk построчно читает файлы и добавляет во временный файл седьмой атрибут (размер файла). Начинаем перебирать файлы в обеих директориях и получаем информацию о дате создания файла. Если файл из эталонной директории новее и имеет такое же название, то мы заменяем файл их обычной директории на файл из эталонной. Если файл был найден в обеих директориях, то мы ничего не делаем, так как мы уже имеем два идентичных файла в обеих директориях. Если же файл присутствует в первой директории, но не присутствует в эталонной, то мы удаляем файл из первой директории.

|  |
| --- |
| Листинг 4.6 Синхронизация содержимого двух каталогов (file7.sh) |
| #!/bin/bash    (cd $1 && ls -Ggh1 | awk '{print $7}') > "tmp1.out"  (cd $2 && ls -Ggh1 | awk '{print $7}') > "tmp2.out"  while IFS= read -r line2  do  lmt2=$(date +%s -r "$2/$line2")  while IFS= read -r line1  do  foundEq=false  lmt1=$(date +%s -r "$1/$line1")  if [[ "$lmt1" < "$lmt2" ]] && [[ "$line1" == "$line2" ]]; then  cp "$2/$line2" $1  break  fi  if [[ "$line1" == "$line2" ]]; then  foundEq=true  break  fi  done < "./tmp1.out"  if [[ "$foundEq" != "true" ]]; then  cp "$2/$line2" $1  fi  done < "./tmp2.out"    while IFS= read -r line1  do  while IFS= read -r line2  do  foundEq=false  if [[ "$line1" == "$line2" ]]; then  foundEq=true  break  fi  done < "./tmp2.out"  if [[ "$foundEq" != "true" ]]; then  rm "$1/$line1"  fi  done < "./tmp1.out"    rm "tmp1.out"  rm "tmp2.out" |

1. Создадим два каталога dir1 и dir2. В обоих каталогах будет лежать файл ex01, но в dir2 он новее. Также в каталоге dir1 лежит файл \*.xlsx, которого нет в dir2, а в dir2 лежит файл \*.docx, которого нет в dir1. В обоих каталогах лежат идентичные файлы ex02.

|  |  |
| --- | --- |
|  |  |
| Рис. 4-11 Состояние dir1 до выполнения скрипта | Рис. 4-12 Состояние dir2 до выполнения скрипта |

|  |  |
| --- | --- |
|  |  |
| Рис. 4-13 Состояние файла ex01 в dir1 до выполнения скрипта | Рис. 4-14 Состояние файла ex01 в dir2 до выполнения скрипта |

1. Произведем синхронизирование двух каталогов, запустим командный файл, эталонным будем считать каталог dir2.



Рис. 4-15 Выполнение скрипта

|  |  |
| --- | --- |
|  |  |
| Рис. 4-16 Состояние dir1 после выполнения скрипта | Рис. 4-17 Состояние dir2 после выполнения скрипта |

|  |  |
| --- | --- |
|  |  |
| Рис. 4-18 Состояние файла ex01 в dir1 после выполнения скрипта | Рис. 4-19 Состояние файла ex01 в dir2 после выполнения скрипта |

В результате выполнения скрипта из директории dir1 был удален файл \*.xlsx, которого нет в эталонной директории dir2. Файл ex01 был заменен на файл ex01 из директории dir2, так как он новее, чем файл в директории dir1. Файл ex02 не был затронут, так как он совпадает с файлом в эталонной директории. Файл \*.docx был добавлен из эталонной директории в директорию dir1, так как его не было в директории dir1.

**Выводы**

В результате выполнения лабораторной работы были получены навыки работы с переменными среды (или переменными окружения) и с возможностями их использования в Linux. Так же были получены навыки работы со скриптовым языком Bash, при помощи которого можно производить командные сценарии и писать наборы полезных для системного администрирования скриптов.