*ПЗ-10:* ***Разработка сценариев оболочки***

**1. Создание сценария оболочки.**

На прошлых занятиях мы разобрали как устроена и какие особенности имеет файловая система (ФС) Linux.

На Лекции-13 рассмотрим **bash-скрипты**. Это — сценарии командной строки, написанные для оболочки **bash**. Существуют и другие оболочки, например — **zsh, tcsh, ksh**, но мы сосредоточимся на **bash**.

**BASH** — **Bourne**-**Again** **SHell** (что может переводится как «перерожденный шел», или «*Снова шел Борн* (создатель **sh**)»), самый популярный командный интерпретатор в юниксоподобных системах, в особенности в **GNU**/**Linux**.

Ниже приведен ряд встроенных команд, которые мы будем использовать для создания своих скриптов:

* **break** выход из цикла **for**, **while** или **until**
* **continue** выполнение следующей итерации цикла **for**, **while** или **until**
* **echo** вывод аргументов, разделенных пробелами, на стандартное устройство вывода
* **exit** выход из оболочки
* **export** отмечает аргументы как переменные для передачи в дочерние процессы в среде
* **hash** запоминает полные имена путей команд, указанных в качестве аргументов, чтобы не искать их при следующем обращении
* **kill** посылает сигнал завершения процессу
* **pwd** выводит текущий рабочий каталог
* **read** читает строку из ввода оболочки и использует ее для присвоения значений указанным переменным
* **return** заставляет функцию оболочки выйти с указанным значением
* **shift** перемещает позиционные параметры налево
* **test** вычисляет условное выражение
* **times** выводит имя пользователя и системное время, использованное оболочкой и ее потомками
* **trap** указывает команды, которые должны выполняться при получении оболочкой сигнала
* **unset** вызывает уничтожение переменных оболочки
* **wait** ждет выхода из дочернего процесса и сообщает выходное состояние.

***Сценарии командной строки*** — это наборы тех же самых команд, которые можно вводить с клавиатуры, собранные в файлы и объединённые некоей общей целью. При этом результаты работы команд могут представлять либо самостоятельную ценность, либо служить входными данными для других команд. ***Сценарии*** — это мощный способ автоматизации часто выполняемых действий.

Итак, если говорить о командной строке, она позволяет выполнить несколько команд за один раз, введя их через точку с запятой: **pwd; whoami**

Если вы опробовали это в своём терминале, ваш первый **bash-скрипт**, в котором задействованы две команды, уже написан. Работает он так. Сначала команда **pwd** выводит на экран сведения о текущей рабочей директории, потом команда **whoami** показывает данные о пользователе, под которым вы вошли в систему

***Командная строка*** — отличный инструмент, но команды в неё приходится вводить каждый раз, когда в них возникает необходимость. Что если записать набор команд в файл и просто вызывать этот файл для их выполнения? Собственно говоря, тот файл, о котором мы говорим, и называется *сценарием командной строки*.

Создайте пустой файл с использованием команды ***touch***: **touch myscript.sh**

В его первой строке нужно указать, какую именно оболочку мы собираемся использовать. Нас интересует **bash**, поэтому первая строка файла будет такой:

**#!/bin/bash** # This is a comment

Команды оболочки отделяются знаком перевода строки, комментарии выделяют знаком решётки. Вот как это выглядит:

|  |  |
| --- | --- |
| **#!/bin/bash**  **# This is a comment**  **pwd**  **whoami** | Сохраните файл, дав ему имя **myscript.sh**, и работа по созданию bash-скрипта почти закончена. Сейчас осталось лишь сделать этот файл исполняемым, иначе, попытавшись его запустить, вы столкнётесь с ошибкой ***Permission denied:*** |
| ***Permission denied:***  *Попытка запуска файла сценария с неправильно настроенными разрешениями* | https://habrastorage.org/getpro/habr/post_images/29f/743/3ec/29f7433ec002900e7518396a814fe351.png |
| **chmod +x ./myscript** | Сделаем файл исполняемым |
| **myscript** | Теперь попытаемся его выполнить |

После настройки разрешений всё работает как надо.

**Вывод сообщений**

|  |  |  |
| --- | --- | --- |
| **echo** | Для вывода текста в консоль Linux применяется команда **echo**. | |
| **#!/bin/bash**  **#** *our comment is here*  **echo "The current directory is:"**  **pwd**  **echo "The user logged in is:"**  **whoami** | | https://habrastorage.org/getpro/habr/post_images/4d0/173/e3d/4d0173e3ddcad01d0790b6b98dd91315.png |
| **#!/bin/bash**  **echo "Hello World!"**  **printf "%s\n" "Hello World"** | | **Hello World!**  **Hello World!** |

А что если надо вывести на экран значок доллара? Попробуем так:

**echo "I have $1 in my pocket"**

Система обнаружит знак доллара в строке, ограниченной кавычками, и решит, что мы сослались на переменную. Скрипт попытается вывести на экран значение неопределённой переменной **$1**. Это не то, что нам нужно. Что делать?

В подобной ситуации поможет использование управляющего символа, обратной косой черты, перед знаком доллара (теперь сценарий выведет именно то, что ожидается):

|  |  |
| --- | --- |
| Использование управляющей последовательности для вывода знака доллара:  **echo "I have \$1 in my pocket"** | https://habrastorage.org/getpro/habr/post_images/771/a8a/710/771a8a7102c7a25f1ecd1e739a00de6e.png |

**2. Использование переменных в файлах сценариях.**

Переменные позволяют хранить в файле сценария информацию, например — результаты работы команд для использования их другими командами.

Нет ничего плохого в исполнении отдельных команд без хранения результатов их работы, но возможности такого подхода весьма ограничены.

Существуют два типа переменных, которые можно использовать в bash-скриптах:

* Переменные среды (Переменные окружения)
* Пользовательские переменные

**2.1 Переменные среды (Переменные окружения)**

Иногда в командах оболочки нужно работать с некими системными данными. Вот, например, как вывести домашнюю директорию текущего пользователя (*При обращении (но не при установке!) к переменной окружения, к ее имени необходимо добавить символ* “$”):

|  |  |
| --- | --- |
| Использование переменной среды в сценарии:  **#!/bin/bash**  **# display user home**  **echo "Home for the current user is: $HOME"** | https://habrastorage.org/getpro/habr/post_images/da0/6b6/4a4/da06b64a489f6a8e16a3ca345f270f82.png |

Обратите внимание на то, что мы можем использовать системную переменную **$HOME** в двойных кавычках, это не помешает системе её распознать. Вот что получится, если выполнить вышеприведённый сценарий.

* **Переменная** **HOME** *указывает на корневой (домашний) каталог пользователя*.

Предположим для примера, что вы пишите сценарий, который должен создать в домашнем каталоге любого пользователя, выполняющего этот сценарий, новый каталог с именем newdir, чтобы установить в нем какой-то пакет программ. Начальные строки файла-сценария могут выглядеть так:

**cd $HOME**

**mkdir newdir**

Тогда любой пользователь, запустивший такой сценарий из любого каталога, получит желаемый результат.

* **Переменная PATH** *содержит в себе список каталогов, в которых система ищет команды или программы для исполнения.*

Другими словами, если пользователь хочет выполнить какую-либо команду (или запустить программу), то он либо должен указать полный путь к команде, либо этот путь должен содержаться в списке переменной **PATH**.

Рассмотрим следующий пример: **echo $PATH**

На экран будет выведено что-то в таком роде: ***.:/home/myfiles/bin:/usr/bin:/bin:/usr/X11R6/bin***

Как можно видеть, каталоги в списке переменной **PATH** разделены знаком “**:**”. В частности, каталог /bin содержит многие из рассмотренных ранее команд системы: cp, gzip, mkdir и другие.

Поэтому мы и можем вызывать такие команды напрямую, без указания пути к ним. Поиск команд или программ ведется по списку каталогов в переменной **PATH** слева направо – об этом необходимо помнить, поскольку в системе и у пользователя могут оказаться команды или программы с одинаковыми именами.

В этом случае будет выполнена команда из того каталога, который расположен в списке раньше.

* Отметим, что текущий каталог, представленный знаком “**.**” в первой позиции списка в данном примере, по умолчанию может и не включаться системными сценариями автозагрузки в состав переменной **PATH**.
* Если это так, то вы не сможете запустить, скажем программу myprog, расположенную в текущем каталоге, просто набрав в командной строке myprog, а должны вызвать ее так: **./myprog**
* (здесь между **точкой** и знаком “**/**” нет пробела).

Другая возможность – ***включить текущий каталог в список переменной*** **PATH**.

Чтобы добавить какой-либо каталог, скажем **/opt/bin**, в список переменной **PATH**, необходимо выполнить команду: **export PATH=$PATH:/opt/bin**

|  |  |
| --- | --- |
| * **Переменная PWD** | *– указывает на текущий каталог* |
| * **Переменная USER** | *– указывает на имя пользователя* |
| * **Переменная EDITOR** | *– указывает на редактор, который будет использоваться по умолчанию некоторыми программами и утилитами* |
| * **Переменная PRINTER** | *– указывает на имя принтера, используемого по умолчанию* |
| * **Переменная TERM** | *– указывает на тип используемого терминала*  Обычно эта переменная имеет значение **xterm** при работе в графическом режиме или **linux** или **vt100** в текстовом режиме работы |
| * **Переменная DISPLAY** | *– указывает на адрес графического дисплея* |

**Переменная DISPLAY** важна, когда необходимо открыть новое **X** окно. При работе на локальном компьютере никаких проблем с использованием переменных **TERM** и **DISPLAY** обычно не возникает. Однако они могут появиться при работе на удаленном компьютере. О том, как правильно установить переменную **DISPLAY**, рассказано в разделе о настройках **Х-окружения**.

Таким образом,

* ***Environment variables***, или ***переменные окружения*** – *это системные переменные, которые используются операционной системой, программами или командными файлами-сценариями*.

Вы можете увидеть, какие переменные окружения установлены у вас и каковы их значения в данный момент с помощью команды **env**.

Значения большинства переменных окружения обычно устанавливают в системных сценариях автозагрузки или сценариях автозагрузки пользователя (см. соответствующий раздел). Более подробно о назначении различных переменных окружения (как и о многом другом) можно прочесть в справочных страницах, посвященных соответствующим командным оболочкам: **man bash**, **man** **tcsh** или **man** **zsh**.

**2.2 Пользовательские переменные**

В дополнение к переменным среды, bash-скрипты позволяют задавать и использовать в сценарии собственные переменные. Подобные переменные хранят значение до тех пор, пока не завершится выполнение сценария.

Как и в случае с системными переменными, к пользовательским переменным можно обращаться, используя знак доллара:

|  |  |
| --- | --- |
| **#!/bin/bash**  **#** *testing variables*  **grade=5**  **person="Adam"**  **echo "$person is a good boy, he is in grade $grade"** |  |

**Подстановка команд**

Одна из самых полезных возможностей bash-скриптов — это возможность извлекать информацию из вывода команд и назначать её переменным, что позволяет использовать эту информацию где угодно в файле сценария.

Сделать это можно двумя способами:

1. С помощью значка обратного апострофа «**`**»
2. С помощью конструкции **$()**

Используя **первый подход**, проследите за тем, чтобы вместо обратного апострофа не ввести одиночную кавычку. Команду нужно заключить в два таких значка:

|  |  |
| --- | --- |
| **mydir=`pwd`** | – Используя первый подход, проследите за тем, чтобы вместо обратного апострофа не ввести одиночную кавычку. Команду нужно заключить в два таких значка |
| **mydir=$(pwd)** | – При втором подходе то же самое записывают таким образом. |

А скрипт, в итоге, может выглядеть так

(*скрипт, сохраняющий результаты работы команды в переменной*):

|  |  |
| --- | --- |
| **#!/bin/bash**  **mydir=$(pwd)**  **echo $mydir** |  |

В ходе его работы вывод команды **pwd** будет сохранён в переменной **mydir**, содержимое которой, с помощью команды **echo**, попадёт в консоль.

**3. Выполнение математических вычислений.**

Для выполнения математических операций в файле скрипта можно использовать конструкцию вида *$((a+b)):*

|  |  |
| --- | --- |
| **#!/bin/bash**  **var1=$(( 5 + 5 ))**  **echo $var1**  **var2=$(( $var1 \* 2 ))**  **echo $var2** |  |

**Сравнение чисел**

В скриптах можно сравнивать числовые значения. Ниже приведён список соответствующих команд.

|  |  |  |
| --- | --- | --- |
| **n1 -eq n2** | Возвращает истинное значение, если n1 равно n2. | |
| **n1 -ge n2** | Возвращает истинное значение, если n1больше или равно n2. | |
| **n1 -gt n2** | Возвращает истинное значение, если n1 больше n2. | |
| **n1 -le n2** | Возвращает истинное значение, если n1меньше или равно n2. | |
| **n1 -lt n2** | Возвращает истинное значение, если n1 меньше n2. | |
| **n1 -ne n2** | Возвращает истинное значение, если n1не равно n2. | |
| **#!/bin/bash**  **val1=6**  **if [ $val1 -gt 5 ]**  **then**  **echo "The test value $val1 is greater than 5"**  **else**  **echo "The test value $val1 is not greater than 5"**  **fi** | | |  |

**4. Выход из сценария.**

Вместо использования **exit 1** (выход из оболочки) попробуйте использовать **return 1**

**5. Редактор vi (vim)**

В редакторе **vi** существует несколько режимов работы.

Наиболее часто используются два из них – командный режим и режим ввода текста. В командном режиме нажатие каждой клавиши клавиатуры означает определенное действие, но не ввод текста. В этот режим пользователь попадает при запуске редактора: **vi filename**

где **filename** - имя редактируемого файла.

В командном режиме vi можно перемещаться по тексту с помощью **клавиш** **управления** **стрелками** (**влево**-**вправо**-**вверх**-**вниз**), а также пользоваться функциями поиска: **/pattern** ищет первое появление образца **pattern** в тексте по направлению вперед, а **?pattern** - по направлению назад.

Можно удалять текст: нажатие клавиши <**X**> удаляет символ, предшествующий курсору, а клавиши <**x**> - символ над курсором.

Стереть текст от позиции курсора до конца строки можно, нажав клавиши **<d><$>**, а стереть всю строку: **<d><d>**.

Чтобы **ПЕРЕйТИ В РЕЖИМ ВВОДА ТЕКСТА**, нажмите клавишу **<i>**.

В этом режиме все, что набирается на клавиатуре, является вводимым текстом. Вернуться обратно в командный режим **vi**, закончив набирать текст можно, нажав клавишу <**Esc**>.

Еще одним режимом редактора vi является так называемый режим последней строки.

Все команды, вводимые в этом режиме, начинаются с символа “**:**”. Ввод двоеточия перемещает курсор в нижнюю часть экрана, где следует ввести оставшуюся часть команды.

Режим последней строки используется главным образом для выполнения команд общего характера, в частности:

* **wq** - записать произведенные изменения в файл и выйти из редактора;
* **w** - записать изменения в файл без выхода из редактора;
* **w** **filename** - записать текущий редактируемый буфер в новый файл с именем **filename**;
* **q** - выйти из редактора;
* **q!** - выйти из редактора, не сохраняя произведенные изменения;
* **!sh** - войти в командную оболочку (**shell**) системы. Здесь вы можете выполнять любые команды оболочки. Чтобы вернуться обратно в редактор, наберите **exit**;
* **!command** - выполнить команду оболочки (командного интерпретатора операционной системы) **command**;

Итак, редактор **vi** имеет три режима:

1. ***Командный*** - в этом режиме можно перемещаться по файлу и выполнять редактирующие команды над текстом. Команды вызываются ОБЫЧНЫМИ ЛАТИНСКИМИ БУКВАМИ.
2. ***Ввода текста*** - в этом режиме обычные латинские буквы будут вставляться в текст.
3. ***Режим строчного редактора*** ED используется для управления файлами (типа сохранить файл, зачитать файл и т.д.)

ПРИМЕРЫ:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **vi имя\_файла** | | | | – vi в КОМАНДНОМ режиме | |
| **ESC : q ! Enter** | | | | – чтобы выйти из файла без сохранения | |
| **ESC : w ! Enter** | | | | – чтобы выйти из файла, сохранив изменения | |
| **ESC : q Enter** | | | | – чтобы выйти из файла, сохранив изменения | |
| **ESC : wq Enter** | | | | – выйти из файла с сохранением, одной командой | |
| для перехода В РЕЖИМ ВВОДА нужно нажать команды типа | | | | | |
| **"i"** | | – вставлять здесь | | | |
| **"A"** | | – вставлять с конца строки | | | |
| **"cw"** | | – заменять текущее слово | | | |
|  | | | | | |
| **ESC** | | | – для ВОЗВРАТА В КОМАНДНЫЙ РЕЖИМ | | |
| **CTRL-[** | | | – для возврата в командный режим | | |
| **":"** | – для перехода В РЕЖИМ УПРАВЛЕНИЯ ФАЙЛАМИ нужно нажать “**:”** (перейти в режим редактора ED) | | | | |
| *Двигаться по файлу можно командами*: | | | | | |
| **h,j,k,l** | | | – влево, вниз, вверх, вправо | | |
| **Ctrl-F** | | | – на страницу вниз | | |
| **Ctrl-B** | | | – на страницу вверх | | |
|  | | | | | |
| **i** | | | – перевод в режим ввода (подгоните курсор к нужному месту и нажмите **i**) | | |
| **ESC** | | | – прекратить ввод, перейти в командный режим | | |
| Подгоните курсор к ненужному месту и нажмите: | | | | | |
| **x** | | | – удалить символ | | |
| **dd** | | | – удалить строчку | | |
| **o** | | | – вставлять с новой строки (под текущей строкой) | | |
| **a** | | | – в режим ввода ЗА курсором | | |
| **5yy** | | | – запомнить 5 строчек | | |
| *Подгоните курсор к нужному месту* | | | | | |
| **p** | | | – вставить запомненные строки под курсором | | |
| **P** | | | – вставить запомненные строки НАД курсором | | |
|  | | | | | |
| **J** | | | – склеить две строки | | |
| **/Шаблон поиска Enter** | | | | | – поиск |
| **n** | | | – повторить поиск | | |

Того, кто желает разобраться с **vi / vim** поподробнее, может предоставить больше удобств и команд по редактированию, отсылаем к прилагаемому справочнику-памятке "***Наиболее употребительные команды vi***", **vibegin.txt** ([***http://lib.ru/unixhelp/vibegin.txt***](http://lib.ru/unixhelp/vibegin.txt)), ну, и, естественно (как всегда), к документации.