## Продвинутая работа с bash

### Цель работы

Продемонстрировать основные навыки работы в командной строке Linux с использованием возможностей интерпретатора bash.

### Задания для выполнения

1. Запускаем командную строку
2. Создаём переменную и присваиваем любое значение





Где export – добавляет переменную окружения tempo



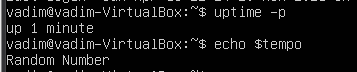
Где source – позволяет выполнить скрипт в текущем командном интерпретаторе, а это значит, что все переменные и функции, добавленные в этом скрипте, будут доступны также и в оболочке после его завершения.

1. Отображаем её



Где echo – отображает переменную (tempo), а $ - отображает содержимое переменной

1. Проверяем, что переменная сохраняет своё значение после рестарта с помощью echo $a



Где uptime -p служит для чтобы выводить время работы системы в минутах (-p значит короткий формат, а именно в минутах)

1. Отображаем переменную $PATH



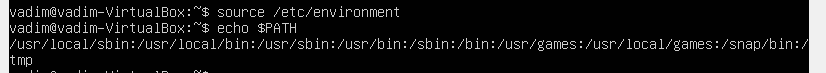
1. Разбираемся с тем как она работает

Изначально, имя исполняемого файла (скрипта) ищется в текущем каталоге, а потом в каталогах, указанных в PATH через двоеточие

1. Делаем так, чтобы добавился ещё один путь — /tmp

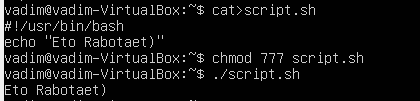
Допишем в конец переменной PATH «:/tmp»





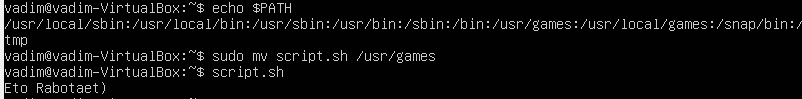
1. Пишем пустой скрипт с расширением .sh и запускаем его. Скрипт - это простой текстовый файл, в первой строке которого содержится шебанг:

#!/usr/bin/bash



./ указывает текущую директорию, в которой находится скрипт

1. Делаем так, чтобы скрипт можно было запускать без указания пути



Где mv обозначает перемещение скрипта в каталог /usr/games (каталог может быть произвольным), это сделано для того, чтобы запускать скрипт без указания пути

**Проверка:**

Запускаем из любой директории:







1. Делаем так, чтобы скрипт инициализировал DATE и задавал значение времени, затем выводил на экран

В редакторе vim:



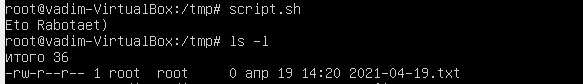
Где date –I обозначает вывод даты в формате год-месяц-число

Проверяем:



1. Делаем так, чтобы вместо вывода на экран создавался файл в директории подобный /tmp/2017-08-17.txt

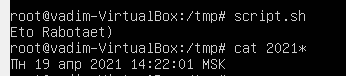




1. Делаем так, чтобы в файл записывалось время создания



Где результат команды date записывается в файл



1. Используя CRON заставляем скрипт исполняться ежеминутно

Cron — это планировщик заданий, который работает на UnixLinux хостинге. Он позволяет автоматически выполнять определенные действия на сервере (запуск программ, скриптов и т.д.), с заданным временем или периодичностью.

Crontab — это команда, используемая для установки, удаления или вывода файла конфигурации cron, используемого для управления демоном cron.

Запускаем скрипт от суперпользователя:



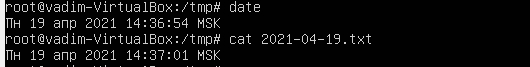
Где crontab –e открывает меню выбора редактора

С помощью vim, дописываем в конце:



Где первая \* - минуты, вторая \* - часы, третья – дни, четвёртая - месяцы и пятая - день недели.

1. Через 10 минут проверяем, чтобы в директории было столько же файлов, сколько прошло минут.



1. Обнаруживаем ошибку

Мы видим, что файл обновляется каждую минуту.

1. Используя find делаем так, чтобы директория проверялась раз в 10 минут и удаляла файлы, созданные больше, чем 17 минут назад





В этом скрипте find осуществляет поиск по директориям

-type f отвечает за поиск именно файлов в каталоге tmp

-mtime +17 отвечает за то, чтобы поиск осуществлялся среди файлов, которые созданы больше чем 17 минут назад

rm принимает результат команды find и удаляет файлы

Используя crontab, заставим скрипт выполняться раз в 10 минут:

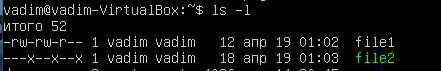


1. Создаём пустой файл и вписываем что-нибудь



1. Копируем с сохранением всех прав в другой файл с любым названием и содержимым







С помощью –p мы можем сохранить права доступа

cp - копирование

1. Сравниваем контрольные суммы

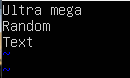
md5sum – это программа, позволяющая вычислять значения контрольных сумм файлов по алгоритму MD5.

Эти контрольные сумму будут одинаковы:



1. Меняем содержимое любого из файлов и смотрим разницу в контрольных суммах





После изменения файла 2, мы видим, что контрольные суммы изменились и теперь различны:



1. Создаём пользователя и задаём ему пароль



-p служит для добавления пароля новому пользователю

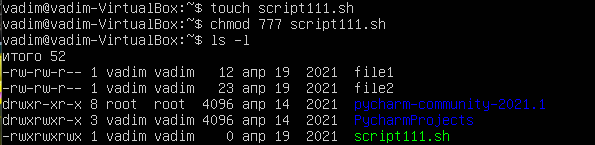
1. Разбираемся с /etc/passwd, /etc/shadow, /etc/group

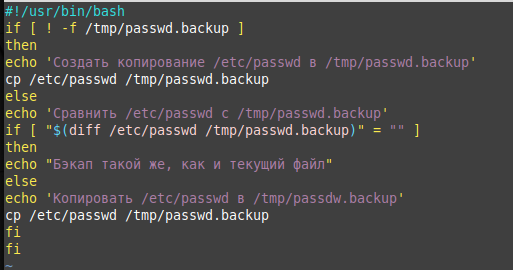
/etc/passwd – содержит информацию о пользователях

/etc/shadow – содержит информацию о защищённых паролях пользователей

/etc/group – содержит информацию о группах

1. Делаем так, чтобы, если нет файла /tmp/passwd.backup, /etc/passwd копировался в /tmp/passwd.backup, а если /tmp/passwd.backup есть, то сравнивался с /etc/passwd





!- обозначает не

-f обозначает проверку существования данного файла

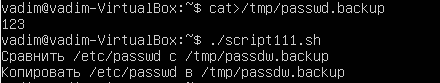
Первый случай, когда бэкап создаётся:



./ указывает текущую директорию, в которой находится скрипт

1. Если различаются, то пусть копируется, если нет, пусть завершает работу скрипта

Второй случай, когда бэкап не соответствует текущему:



Третий случай, когда бэкап был такой же, как и текущий файл:

