Bash Scripting

Lesson 1

- **Creating a first shell script**

Why Scripting in Bash Makes Sense

- /bin/sh is the default shell since the very first days of UNIX
  - Many alternative shells have been developed and are still available and used
  - /bin/bash is the most common shell on Linux and other Unices
- Large parts of the operating system and its applications are written in bash
  - Look for instance in /etc/init.d, /etc/profile and many more

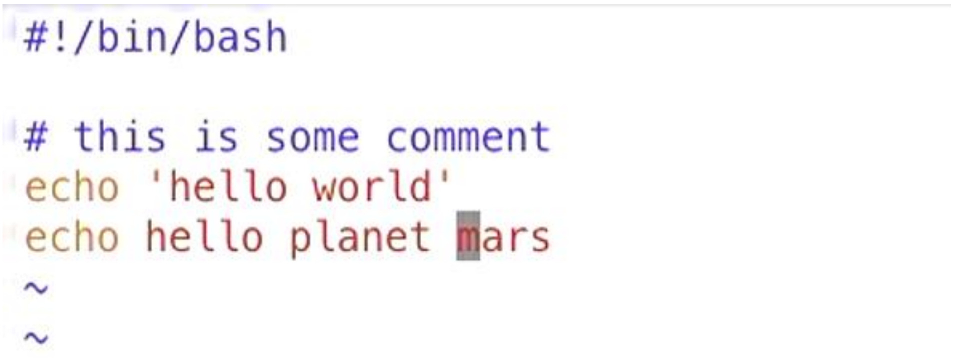
What exactly is a script?

- A script is a simple program that doesn't have to be compiled
- A script can be a mere list of commands that are executed sequentially
- Or more clever things can be accomplished, using smart elements like:
  - variables
  - conditional structures
  - user input processing

Choosing an editor

- Any text editor will do
- **vim** is a popular editor on Linux
- Once **vim** knows which content is in a file, syntax highlighting is used
  - Tip! Start your script with the line #!/bin/bash, close the document, and open it again to get syntax highlighting

Eg:



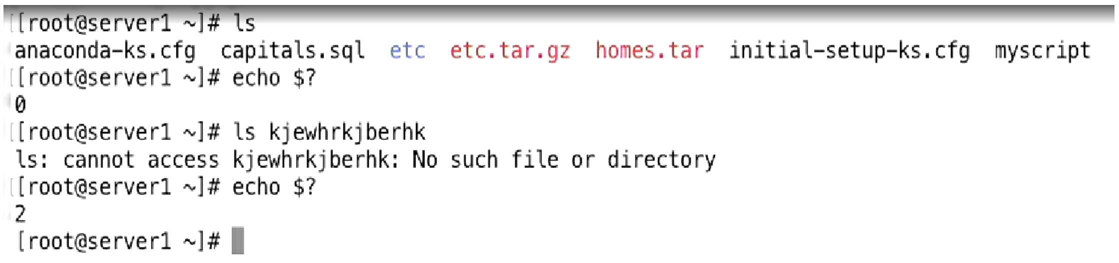
```
#!/bin/bash

# this is some comment
echo 'hello world'
echo hello planet mars
~
~
```

Core bash script ingredients

- The "shebang"; #!/bin/bash
  - # is NOT a comment sign in this case
- Comment lines explaining what the script wants to accomplish
- White lines and other structural components that make the script readable
- An **exit** statement if you don't want to use the exit status of the last command in the script

Eg for exit code :



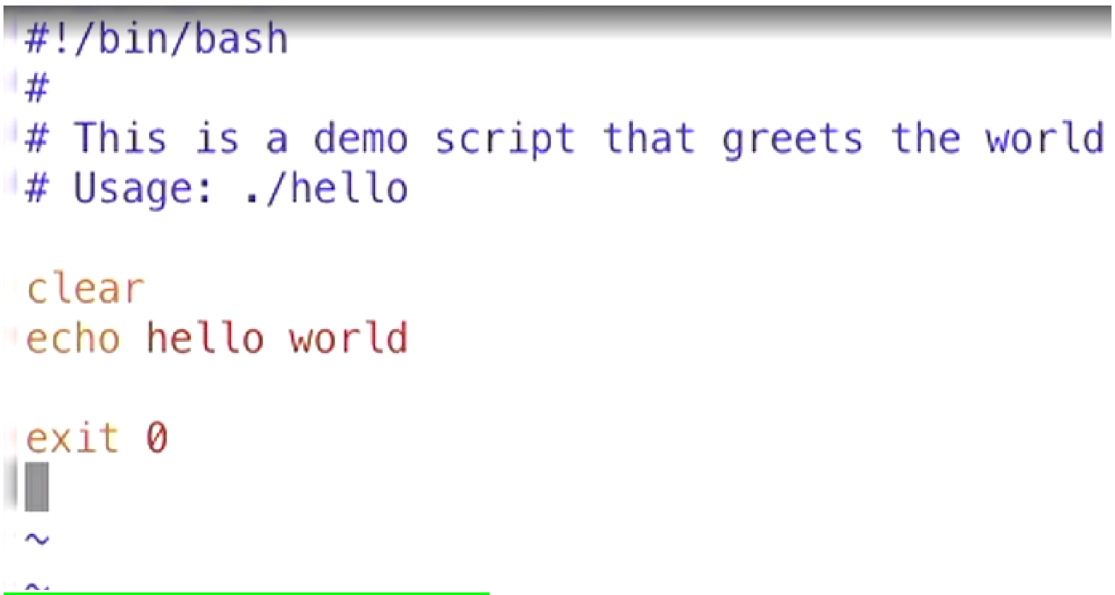
```
[root@server1 ~]# ls
anaconda-ks.cfg  capitals.sql  etc  etc.tar.gz  homes.tar  initial-setup-ks.cfg  myscript
[root@server1 ~]# echo $?
0
[root@server1 ~]# ls kjewhrkjberhk
ls: cannot access kjewhrkjberhk: No such file or directory
[root@server1 ~]# echo $?
2
[root@server1 ~]# 
```

Example scipt:

```
#!/bin/bash
#
# This is a demo script that greets the world
# Usage: ./hello

clear
echo hello world

exit 0
```



```
#!/bin/bash
#
# This is a demo script that greets the world
# Usage: ./hello

clear
echo hello world

exit 0

~
```

The above bash file contains:

1st line == shebang which indicated or redirected which interpreter should I use.
2nd line == contains some comment for better understanding of our script
3rd line == have a white line which help us to readable more
4th line == have some follow up commands
5th line == white line
6th line == have a exit status if the script works properly it return the 0 and indicated all went well if it returns 1 have error and other codes have a specific error according to the condition.

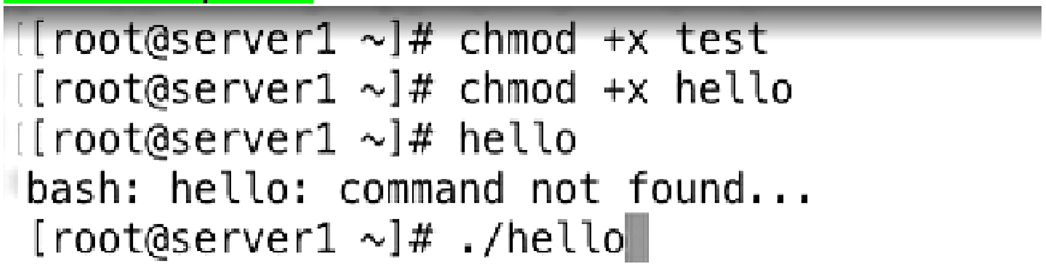
Storing and running the script

- On Linux the current directory is not in the $PATH variable
- Consider storing the script in a directory that is in $PATH, such as /usr/local/bin or $USER/bin
- Or run the script using **./myscript**
- Notice that running a script without **./** in front may lead to unexpected results

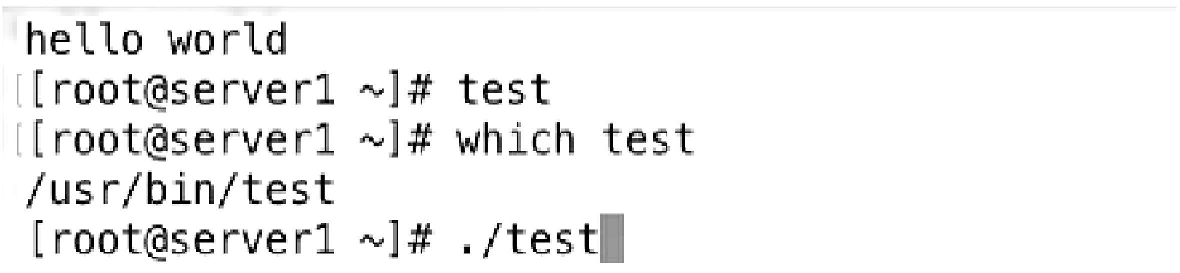
Make your script is executable

- In order to run the script, it must have the Execute permission applied
    - **chmod +x myscript**
- If the script is started as an argument to the bash shell, it doesn't need the execute permission itself
    - **bash myscript**

Lets do some practice

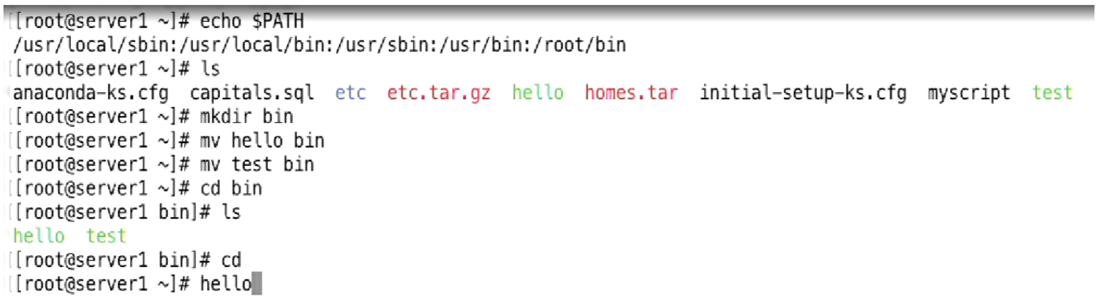


```
[root@server1 ~]# chmod +x test
[root@server1 ~]# chmod +x hello
[root@server1 ~]# hello
bash: hello: command not found...
[root@server1 ~]# ./hello
```

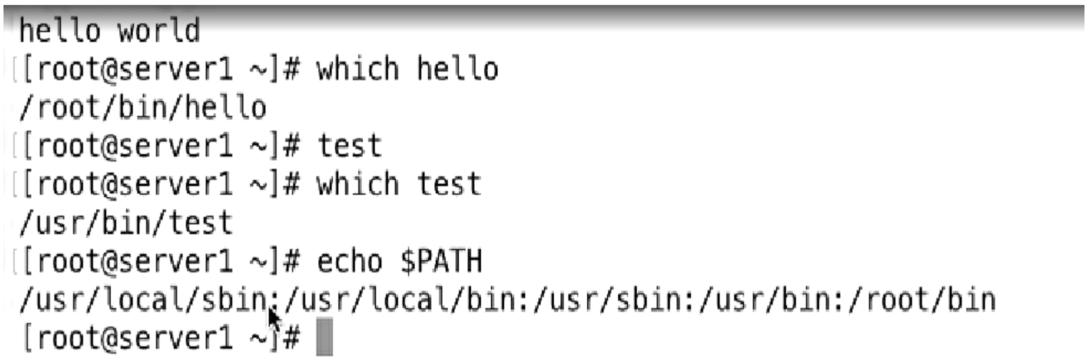


```
hello world
[root@server1 ~]# test
[root@server1 ~]# which test
/usr/bin/test
[root@server1 ~]# ./test
```

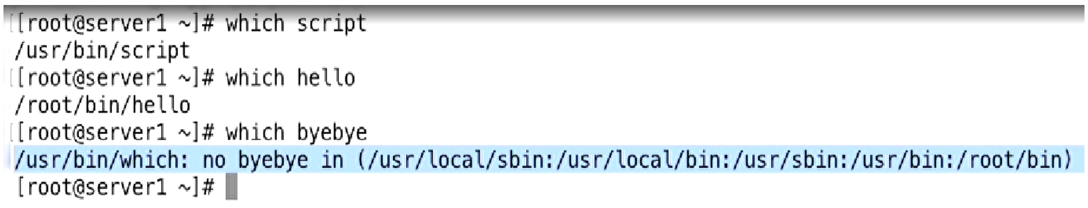
/usr/local/sbin is the place to store the system binaries
/usr/local/bin is the best place to store the custom scripts and need to available for everybody



```
[root@server1 ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
[root@server1 ~]# ls
anaconda-ks.cfg  capitals.sql  etc  etc.tar.gz  hello  homes.tar  initial-setup-ks.cfg  myscript  test
[root@server1 ~]# mkdir bin
[root@server1 ~]# mv hello bin
[root@server1 ~]# mv test bin
[root@server1 ~]# cd bin
[root@server1 bin]# ls
hello  test
[root@server1 bin]# cd
[root@server1 ~]# hello
```



```
hello world
[root@server1 ~]# which hello
/root/bin/hello
[root@server1 ~]# test
[root@server1 ~]# which test
/usr/bin/test
[root@server1 ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
[root@server1 ~]#
```

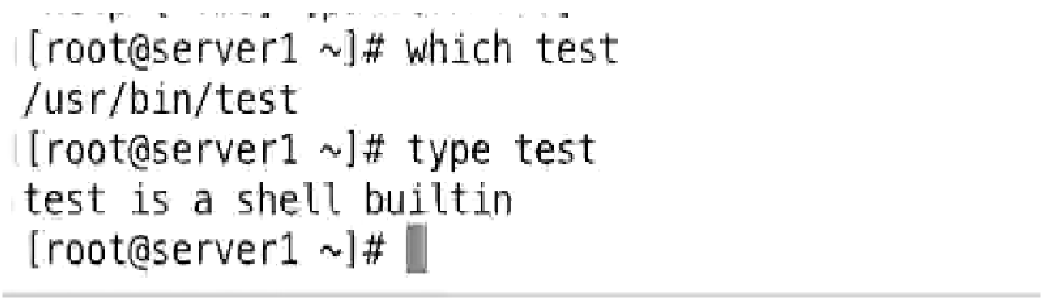


```
[root@server1 ~]# which script
/usr/bin/script
[root@server1 ~]# which hello
/root/bin/hello
[root@server1 ~]# which byebye
/usr/bin/which: no byebye in (/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin)
[root@server1 ~]#
```

Using bash internal commands versus external commands

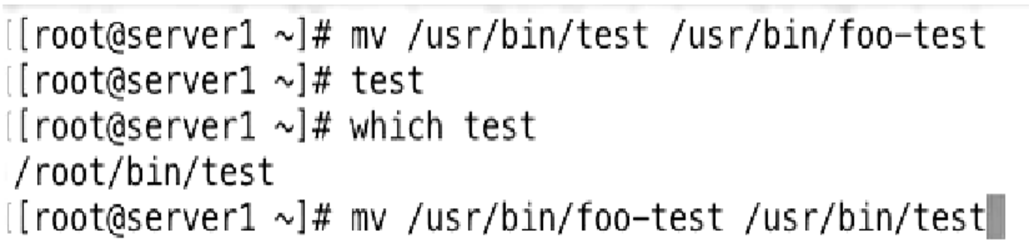
- An *internal command* is a part of the Bash shell
  - It does not have to be loaded from disk and therefore is faster
  - Use **help** to get a list of all internal commands
- An *external command* is a command that is loaded from an executable file on disk
- External commands normally are slower

Type help you to get to know about it is internal or not?



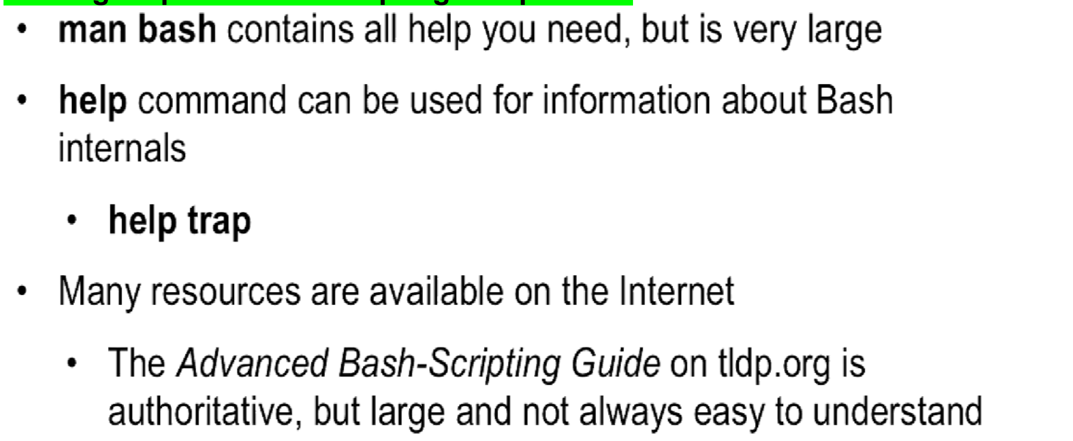
```
[root@server1 ~]# which test
/usr/bin/test
[root@server1 ~]# type test
test is a shell builtin
[root@server1 ~]#
```

Bash first executes the internal command then it execute the external command if the internal command is not present there.

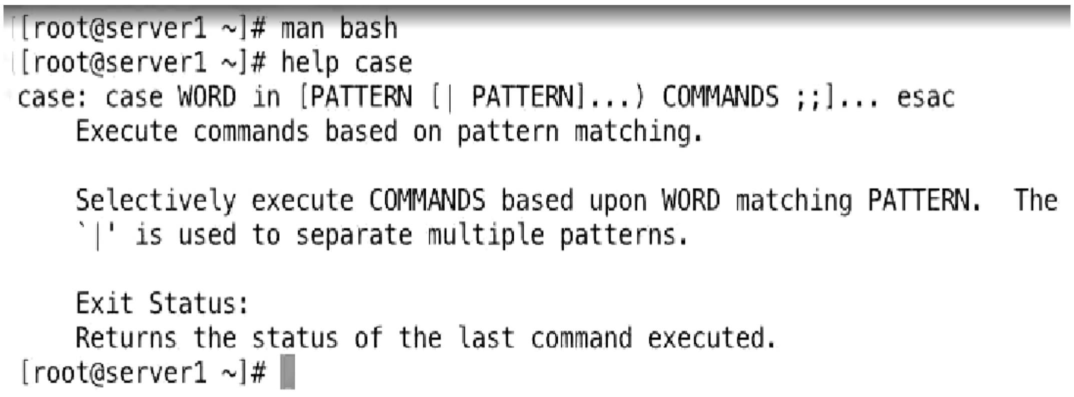


```
[root@server1 ~]# mv /usr/bin/test /usr/bin/foo-test
[root@server1 ~]# test
[root@server1 ~]# which test
/root/bin/test
[root@server1 ~]# mv /usr/bin/foo-test /usr/bin/test
```

Finding help about the scripting components



- **man bash** contains all help you need, but is very large
- **help** command can be used for information about Bash internals
  - **help trap**
- Many resources are available on the Internet
  - The *Advanced Bash-Scripting Guide* on tldp.org is authoritative, but large and not always easy to understand

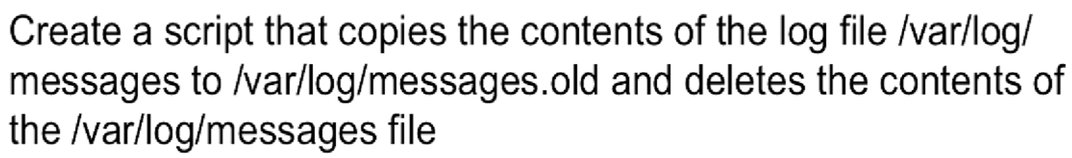


```
[root@server1 ~]# man bash
[root@server1 ~]# help case
case: case WORD in [PATTERN [| PATTERN]...) COMMANDS ;;]... esac
    Execute commands based on pattern matching.

    Selectively execute COMMANDS based upon WORD matching PATTERN.  The
    `|' is used to separate multiple patterns.

    Exit Status:
    Returns the status of the last command executed.
[root@server1 ~]# 
```

Exercise 1



Create a script that copies the contents of the log file /var/log/messages to /var/log/messages.old and deletes the contents of the /var/log/messages file

Solution 1

```
#!/bin/bash
# This script copies /var/log contents and clears current
# contents of the file
# Usage: ./clearlogs

cp /var/log/messages /var/log/messages.old
cat /dev/null > /var/log/messages
echo log file copied and cleaned up

exit 0
```