Linux Administration

Scripting

Xavier Belanger

Shell script basics

- A script contains various shell commands to be executed in an automatic fashion.
- This is a text file, no compilation needed.
- To be interpreted properly, the first line should contain a specific command referencing the shell command to use (also called “sh-bang” or “shebang”):

```
#!/bin/bash
```

- Lines beginning with a hash symbol (#) are comments.
- The file must have the execution bit set (“*chmod +x myscript.sh*”).

Saving a script file

- Files can be copied to various locations depending on their purpose:
  - $HOME/bin for your personal script collection [1]
  - /usr/local/bin for scripts shared with all users on the system.
  - /usr/local/sbin for scripts to be used by root only.
- If the script is in your $PATH, you can call it by name directly. Otherwise, you will need to prefix the file name with the dot and slash characters (*./myscript.sh*).
- There is no need for a file extension; you can create a script with or without .sh in the name.

1: this would require to adjust the $PATH variable

Using regular commands

- Command available on the system can be integrated into a script (usually excluding the interactive ones).
- You may want to call commands with their fullpath name (“*/usr/bin/date*” instead of “*date*”) to avoid possible conflicts and aliases issues.
- When using specific option, the long format may be recommended; if using the short format a comment may be needed.

Variables

- To set a variable, directly use a name with the value assigned:

  variable=”hello world”

- Calling a variable is done by using its name prefixed with a dollar sign

  echo $variable

- Variable names are case sensitive, and can only include letters, number and the underscore character.
- By default, all variables are strings. You can declare an integer by using the keyword “let”:

  let variable=5
  echo $(($variable+5))

Test constructs

- To perform a logical test you can use the keyword “test” or the alias “[“:

```
test -e /etc/passwd
[ -e /etc/passwd ]
```

- For arithmetic testing you should use a different syntax:

```
(( 5 > 4 ))
```

- Spaces are important!

Common file test operators

- *-e file* - the file exists
- *-f file* - the file is a regular file
- *-s file* - the file size is not null
- *-d file* - the file is a directory
- *-r/-w/-x file* - the file has the appropriate permission for the user running the test

Common string test operators

- *str1 == str2* - string 1 equals string 2
- *str1 != str2* - string 1 doesn't equals string 2
- *-z str1* - string 1 is null
- *-n str1* - string 1 is not null

Common integer test operators

- *-eq* - equal
- *-ne* - not equal
- *-gt* - greather than
- *-ge* - equal to or greater than
- *-lt* - lesser than
- *-le* - equal to or lesser than

Combining tests

You can perform multiple tests at once using -a for a logical AND or -o for a logical OR:

- <test1> -a <test2>
- <test1> -o <test2>

if condition

if <test>

then

    <code>

else

    <code>

fi

for loop

for argument in list

do

<*code*>

done

Each item in the list must be delimited by spaces.

while loop

while <test>

do

<code>

done

References

- One of the most complete references about Bash scripting is the “Advanced Bash-Scripting Guide”, available online.
- https://tldp.org/LDP/abs/html/
- You can also use the ShellCheck tool to find bugs in your code:
- https://www.shellcheck.net/

Annex: updating the $PATH variable

- Edit your .profile or .bashrc file
- Add the following lines:

```
PATH=$PATH:$HOME/bin
export PATH
```

- This will take effect at your next login.