Linux Administration

Scripting

Xavier Belanger

This work is licensed under
a Creative Commons Attribution-ShareAlike 4.0 International License.

https://creativecommons.org/licenses/by-sa/4.0/

You are free to:

- **Share** — copy and redistribute the material in any medium or format
- **Adapt** — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

- **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Shell script basics

- A script contains various commands to be executed in an automatic fashion.
- A script is just a text file, no compilation needed.
- To be interpreted properly, the first line should contain a reference to the shell 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).
- A file extension is not required; you can create a script with or without .sh in the name.
- Like for any other programming or scripting development, using a version control system, such as Git, is recommended.

1: this would require to adjust the $PATH variable, see appendix

Using regular commands

- Commands available on the system can be integrated into a script (usually excluding interactive ones).
- You may want to call commands with their full path 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))

Script arguments

Any argument used with a script can be used as a variable:

- $0 is the name of the script
- $1 is the first argument, $2 the second one, and so on.
- $# is the number of arguments.

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 corresponding 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* - greater 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 *&&* for a logical AND or || for a logical OR:

- [test1] && [test2]
- [test1] || [test2]

if condition

```
if [ -r /tmp/tempfile.txt ]
then
        /usr/bin/cat /tmp/tempfile.txt
else
        /usr/bin/printf “Missing file\n”
fi
```

for loop

for user in $(/usr/bin/grep sh$ /etc/passwd | /usr/bin/cut -d ":" -f 1)

do

/usr/bin/last $user

done

Each item in the list must be delimited by spaces or new lines.

while loop

```
/usr/bin/cat server-list | while read ipAddress
do
        /usr/bin/ping -c 2 $ipAddress
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/

Appendix: updating the $PATH variable

- Create a $HOME/bin directory
- Edit your .profile or .bashrc file
- Add the following lines:

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

- This will take effect at your next login