Seneca | SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

Unix Scripting

Lecturer: Shahdad Shariatmadari

August 2020

Agenda

- Some discussion about
  - the Expansions

Lets have some practice on Expansions!

- Brace expansion
- Tilde expansion, parameter and variable expansion, command substitution, arithmetic expansion
- Word splitting
- Pathname expansion

What is the difference between `$name` and `${name}`

- Try this:
  - `name=seneca`
  - `echo $name`
  - `echo $name.txt`
  - `echo $name_txt`
  - `echo ${name}_txt`
  - `echo “${name}”_txt`
- Try this:
  - `name=“seneca     college”`
  - `echo $name`
  - `printf “%q\n” $name`
  - `printf “%q\n” “$name”`
- What have you observed?

Why to quote variables?

- This is one of the best practices in shell scripting to quote variables in order to maintain placeholders.
- Try this:
  - `printf “%q\n” me $YOU us`
  - `printf “%q\n” me “$YOU” us`
- What have your observed?

Which of the following commands are successfully executed?

- $-> `a=1  b=2`
- $-> `a=1  date`
- $-> `a=1  date whoami`
- $-> `date  a=1`

set/unset

- set
  - set a shell variable
  - `set value = 7`
- unset
  - delete a shell variable
  - `unset value`
- setenv:
  - set an environment variable
  - `setenv PATH ${PATH}:$HOME/a bin`

set command

- `set -x` enables a mode of the shell where all executed commands are printed to the terminal
  - One of the typical use case for `set -x` printing every command as it is executed may help you to visualize the control flow of the script if it is not functioning as expected.
  - `set +x` disables it
  - To know more about set, run the following commands:
    - `type set`
    - `help set`

set command

- option "e" makes the shell script error out whenever a command errors out. It's generally a good idea to have it enabled most of the time.
- option "x" makes the shell print out commands after expanding their parameters but before executing them. Useful when debugging but can get overwhelming sometimes.
- More about set:
  - https://www.gnu.org/software/bash/manual/html_node/The-Set-Builtin.html

seq command

- **seq** - generates sequences of numbers:
  - `seq 5`
  - `seq 5  8`
  - `seq 1.3 1.5 10`
- What is the output of the following command?
  - `echo seq{1..10..2}`

What does the output of executing the following commands?

- `touch {a..c}{a..c}{a..c}`
- `echo aa*`
- `echo a{a,b}*`

What does the output of executing the following commands?

- Run this command:
  - `myfunc() { printf ″%q\n″ $*; }`
- Call the function as follow:
  - `myfunc 1 2 3`
  - `myfunc “1 2″ 3`
- Change the function as follow and call it again
  - `myfunc() { printf ″%q\n″ “$*″; }`
- Change the function as follow and call it again
  - `myfunc() { printf ″%q\n″ “$@″; }`
- What have you observed?

Other form of shell expansions

- echo $$
  - Process ID of the Shell
- echo $!
  - Process ID of the most recent command (or background command)
- echo $_
  - Last argument to the previous command

Try this

- A:
  - `sleep 30 &`
  - `echo $!`
  - `kill $!`
- B:
  - `echo Hello World`
  - `echo $_`

Practice parameter and variable expansion

- Try the following commands:
  - `Fname=(joe bob mary sue)`
  - `D eclare -p Fname`
  - `printf “%q\n” “${Fname[*]}”`
  - `printf “%q\n” “${Fname[@]}”`
  - `printf “%q\n” “${!Fname[@]}”`
  - `printf “%q\n” “${!Fname[*]}”`

- What have you observed?

Practice parameter and variable expansion

- Try the following commands:
  - `printf “%q\n” hello “${FOO?Enter Value}”`
  - `FOO=`
  - `printf “%q\n” hello “${FOO?Enter Value}”`
  - `printf “%q\n” hello “${FOO:?Enter Value}”`
  - `printf “%q\n” hello “${COLOR:-blue}”`

- What have you observed?

Class activity

- Complete all the activities in the previous slides and explain what have you observed.
- Submit the word file which contains your answer, matrix screenshot.