Shell Scripting

Introduction

- Shell scripting with bash is very powerful. Many programs / tools / utilities are coded in shell at begining of project and later on same code in written in C, C++ etc. for efficiency after project turns out to be useful.

When not to use shell scripting

- CPU intensive processing
- Programs that require complex math
- Security is very important
- Mission critical applications
- GUI programs
- Access to hardware like Joystick, Camera etc. is required
- Closed source applications :)

Creating scripts

- #! (sha bang)
- Change the mode of the program – 'chmod +x'
- We can pass them as command line arguments. No need to change the mode in this case.

Hello Script

Uncommon Interpreters

- Any program that can take filename as input and does something with that file can be chosen as interpreter.
- e.g. -- cat, more, less, grep etc.

Variables

- We don't need to declare variables. Similar to other scripting languages like javascript, python.
- They don't have any type.
- If we try to use shell variable without using any values, then it is treated as null
- Use $ to access value. No need of $ while setting value.
- These are case sensitive.

Quotes

- Single Quotes(' ')
- Double Quotes (“ ”)

Arrays

- We have arrays which can take integral indexes.
- Even non-array variables are treated as arrays. Just access the 0th element.
- To access arrays-- ${<variable>[<index>]}

Taking Input

- read keyword
- $ read <variable>
- There are many options for read
- -a, -t, -n, -s
- What if variables are more?
- What if values are more?

Special Characters

- Hash (#)
- Semi-colon (;)
- Dot(.)
- Double-quote(“)
- Single-quote(’)
- Comma(,)
- Back-slash(\)

Special Characters

- Back-tick (`)
- Colon(:)
- Asterisk(*)
- Question mark(?)
- Hypen(-)
- Tilde(˜)

Hash (#)

- Used for comments in shell
- Can be used in command lines as well

Semicolon (;)

- Used to specify more than one commands in the same line.
- Used in if, case and while – to be done later.

Dot (.) and quotes (“ and ')

- “ ' are used for grouping strings. “ treats $ as special character, whereas, ' treats $ as normal character.
- . means current directory and .. means parent directory.
- . followed by a command means source script. In this case the script will run in the same environment and shell.

Comma (,)

- Comma can be used in arithmetic expressions. All the expressions are evaluated, but only the last value is returned. Similar to C.
- It can also be used to combine strings in shell. e.g. -- 'a{b,c}d' evaluates to 'abd acd'

Backslash (\)

- Used to escape special characters.
- Can also be used as line continuation.
- Can be used on command line as well.

Back-tick (`)

- Can be used to give command line arguments to other commands
- Can also be used to capture output to a variable for further operation.

Colon (:)

- Shell synonym for true
- It can be used as null statements in if
- It can be used to create anonymous here documents (may be used for multiline comments)
- It can be used to truncate / create files.

Asterisk (*)

- Wildcard to match anything including null
- Multiplication operator
- ** can be used for exponentiation

Question Mark

- Used as ternary operator in mathematical expressions.
- Wild card to match single character.
- Can be used to check if some variable is set or not.

Hyphen (-)

- Hyphen can be used to go to $OLDPWD when supplied to cd
- Arithmetic Subtraction
- Used to pass various type of command line arguments
- Can be used to supply default values to parameters
- '--' end of command line arguments.

Hyphen as stdin / stdout

- Cat - > test.txt
- Cat test.txt - > newtest.txt
- Grep abc newtest.txt | diff test.txt -

Tilde (~)

- Can be used to find home folders of various users
- It can be combine with + or - to find PWD or OLDPWD respectively.

Parameter Substitution - ${}

- Parameter substitution can be used to get values of variables
- It can be used to check if variable is set / not set and change / use values accordingly.
- It can be used to remove longest/shortet pattern match from begining / end of value
- It can be used for search/replacement.
- It can also be used for extracting substrings from variables

Special Variables

- FUNCNAME, CDPATH, SECONDS
- $1, $2, $3 ...
- $@, $*
- $_, $?, $$

Next class...

- If – else
- Case
- loops