Bash Scripting Tips

Shell types

- Bourne, ksh, csh, bash, dash, "sh"
- Portable shell scripts
- Posix compliant shell "--posix"

Bash script execution

- Significance of "#!"
- Bash libraries
- No aliases in script (default)
- set -u : script exits if any uninitialized variable is used
- set -e : script exits if any condition returns false

Bash login shell

- bash login shell:
- executes /etc/profile & one of the following in the given order (not all!)
- ~/.bash_profile
- ~/.bash_login
- ~/.profile
- when login shell exits it runs ~/.bash_logout (if it exists)

Bash non-login shell

- BASH non-login shell
- executes ~/.bashrc
- --norc - disables reading of rc file

Bash command execution

- Alias, function, built-in, external command (via PATH)
- Which command
- Type command

Using only built-ins

- change dir : cd dir
- list files : echo *
- cat files : while read line ; do echo $line; done < file
- create a file : echo "Sample text" > file.txt
- copy src dst : while read line; do echo $line; done < file.txt > newfile.txt
- delete contents : > /tmp/file.txt
- reboot system : echo 1 > /proc/sys/kernel/sysrq followed by

  echo b > /proc/sysrq-trigger
- edit a file :(limited to line)
- while read line; do

```
if [[ "$line" =~ .*STRING_FIND.* ]]; then
     echo "STRING_REPLACEMENT"
   else
    echo $line
 fi
done < /tmp/resolv.txt > /tmp/newfile.txt
```

Debugging bash scripts

- Log to console, file
- Error, warn, info, debug levels
- Set options
  - -v, -x, bash -vx, set -vx, set +vx
- PS2
- Basic syntax check : bash -n
- Debugging using system commands – to be covered later.

PS: prompt statement

- Export PS1="\u@\h \w>"
- Export PS2="continue->"
- PS3 = assignment
- PS4 --- used by set -x
- PROMPT_STATEMENT – displayed before every PS1 display

Scripting guidelines

- Library and main script
- Indentation: space/tab
- Split big functions, big lines
- Variables – small case, underscores, use limited caps
  - Initialize them, use quotes & braces
- Dont use too much advanced features that hinder readability and maintainability
- Function names- small case,underscores, name to reflect the intention – no need of comment when calling the function

Scripting guidelines

- Fail early – do all needed checks at begining
  - Input Args(including functions), critical resources, commands etc
- Exit meaningfully
  - Provide necessary information via log, exit code for the caller (127+)
- Trap signals as necessary
- Console output to be meaningful, clear, indented and as little as possible (log to file for details).
- Progress messages for long operations

Scripting guidelines

- Script names, extension for exe & lib
  - Many opensource exes in /bin , /sbin etc are shell scripts
- Comments must for non trivial function definition, complex logic

Scripting Guidelines

- Use $(command) instead of backticks ` `
- Use [[ ]] instead of []
  - Regexp enabled, pathname expansion disabled
- Wildcard expansion of filenames in scripts
- Use local variables wherever possible
- Declare -r <var> - for read only vars
- Always check for return values from functions
- Use built-in instead of external command (($X+ $Y)) will do instead of $(expr $X + $Y)
- Use common sense and be consistent

For loops

- for i in {1..5}; do echo $i; done BASH_VERSION=3+
- for i in {1..10..3}; do echo $i; done BASH_VERSION=4+ includes skipval as well
- for ((i=0; i< 10; i+=2)); do echo $i; done
- for ((; ;)); do--done - infinite loop
- for i in *; do echo $i; done - list files in current directory
- for i in /etc/*; do echo $i; done - list files in /etc

Bash misc....1

- z=`expr $z + 3`
- z=$((z+3)) (no need of $z)
- let z=z+3
- let "z = z + 3" (spaces allowed in quotes)
- ((z+=10))

- Length of variable : x=LTEMGR; echo ${#x}

Bash misc..2

- Tr -d " " --> use for input arguments to truncate white space
- Bash indirection: message=hello; hello=goodbye; echo ${!message}
- Output redirection : &>
- Background execution -- &
- $$ - current pid, $! - last bkground pid

Bash var expansion

- ${#foo} Number of characters in (length of) foo
- ${foo:3:5} Characters 3 through 5 of foo
- ${foo:4} Foo beginning from the fourth character (chars 4 through end)
- ${foo#STRING} Foo, but with the shortest match of "STRING" removed from the beginning
- ${foo%STRING} Foo, but with the shortest match of "STRING" removed from the end
- ${foo%%STRING} Foo, but with largest match of "STRING" removed from the end
- ${foo##STRING} Foo, but with largest match of "STRING" removed from the beginning
- ${foo/bar/baz} Foo, but with first occurance of string "bar" replaced by string "baz"
- ${foo//bar/baz} Foo, but with all occurances of string "bar" replaced by string "baz"

- ${foo:-bar} If foo is unset, substitute the value "bar" of instead
- ${foo:-$bar} If foo is unset, substitute the value of variable bar instead
- ${foo:=bar} If foo is unset, substitute the value bar and set foo=bar