**Bash Shell Scripting Quick Reference**

**IF Statements**

Checking a file or directory:

**-r** / **-w** readable / writable file

**-x** / **-f** executable / ordinary file

**-e** / **-s** file exists / file size greater than 0

**-d** *file* file is a directory

*if [* ***!*** *-s file ] ; then . . . else . . . fi*

Checking strings:

*s1* **=** *s2* s1 equals s2.

*s1* **!=** *s2* s1 is not equal to s2.

**-z** *s1* s1 has size 0.

**-n** *s1* s1 has nonzero size.

*s1* s1 is not the empty string.

*if [[ “$var” == "hello" ]]; then . . . fi*

Checking numbers:

**-eq**/**-ne, -lt**/**-le, -gt**/**-ge** m ==, != , < , <= , > , >= n

*if [ $x –eq $x ]* check if x is an integer

Checking with command result:

*if* ***grep -q shell bshellref***

Boolean operators:

**!** / **-a** / **-o** not / and / or

*if [ $num -lt 10 -o $num -gt 100 ]*

*if test \( -r $file1 -a -r $file2 \) -o \( -r $1 -a -r $2 \)*

Case statement:

**case** "$var" in

a) cmd1;;

b) cmd2;;

\*) cmd3;; // if all others are not matched, it comes here.

**esac**

**Variables and Values**

Built-in Variables:

$0, $n name of the program, the n-th argument

$# number of command line parameters

$\*, $@ all of the command line parameters

$- options given to the shell

$? return the exit status of the last command

$$ process ID of shell running the script

Quoting:

\c take character c literally

`cmd`, $(cmd) run cmd and replace with its output

"whatever" take as is, after first interpreting $, `...`, \

'whatever' take whatever absolutely literally

*var=`ls \*.bak`* put names of .bak files into variable var

*echo "$1$2hello"* print value of $1 and $2 and string hello

*echo ${abc}\_xyz* print value of $abc, appended with \_xyz

*${!var}* indirect variable referencing

*chmod 755 $(find . -type d)* use cmd output as input list

Arithmetic: uses long integers, usually with **$[**…**]**

Operators in order of precedence:

**\*** / **%** (times, divide, remainder)

**+ -** (add, subtract)

**< > <= >=** (the obvious comparison operators)

**== !=** (equal to, not equal to)

**&&** (logical and)

**||** (logical or)

**=** (assignment)

*result=$[$1 + 3]*

*result=$(expr $2 + $1 / 2 + $3 \\* 5)* (note the \ on the \* symbol)

*var="${var:=ABC}"* (var=”ABC” if unset or empty)

*: ${var:=ABC}* (same as above}

**Loop Statements**

for/while/until loop structure

**for** [condition]; **do** command; commands; *break*; **done**

**while/until** [condition]; **do** command; commands; *continue*; **done**

**while read line**; **do** command; *eval* $cmd; commands; **done <** *$infile*

for loops

*for number in $nlist for number in 1 2 3*

*for file in \*.tar.gz for x in `ls -tr \*.log`*

*for i in {1..5}* // Bash 3.0+ *for i in {0..10..2}* // Bash 4.0+

*for (( i=1; i<=$num; i++ ))*

select loop structure

options=“opt1 opt2 ”; **select** opt **in** $options; **do** commands; **done**

**Parameter Expansion**

String trimming: F = ~/temp/records/example.txt”

${F##\*/} => example.txt

${F#\*/} => temp/records/example.txt

${F%%/\*} => ~

${F%/\*} => ~/temp/records

**Command Execution**

cmd1 && cmd2 Run cmd1, only if successful, run cmd2

cmd1 || cmd2 Run cmd1, only if not successful, run cmd2

cmd1; cmd2 Run cmd1, after finished, run cmd2

cmd1 & cmd2 Run cmd1, start cmd2 immediately

(cmds) Run cmds (commands) in a sub-shell

**Operations**

CentOS 7.0

<**ctrl**-**r**> search matching command, reversely

<**ctrl**-**l**>, **clear** clear window display

<**ctrl**-**p**>/<**ctrl**-**n**> previous/next command

<**alt**-**`**> switch terminal window within workspace

<**ctrl**-**alt**-(**←**,**→**)> switch workspace

<**ctrl**-**shift**-**alt**- (**←**,**→**)> move window to another workspace

Read from keystrokes

read num

I/O Redirection:

pgm > file pgm output redirected to file

pgm < file pgm reads input from file

pgm >> file pgm output appended to file

pgm1 | pgm2 pgm1 output piped into pgm2

n>file stream n output redirected to file

n>>file stream n output appended to file

n>&m stream n output merged with stream m

n<&m stream n input merged with stream m

<<tag standard input comes from here through tag

File descriptor (stream) **n**:

0/1/2 standard input/output/error output

*./script.sh > /dev/null 2>&1* suppress standard output and error

**Array Variables**

*arr=()* initialize an array *arr*

*${arr[n]}* array element *n* (starting at *0*)

*${arr[\*]}* all of the items in the array

*${!arr[\*]}* all of the indexes in the array

*${#arr[\*]}* number of the items in the array

*declare error\_${code}=3* assign a value to a variable name with ref