Cheatography

Shell Cheat Sheet

by Torvak via cheatography.com/32041/cs/9845/

Variables

`var=value;`	Initialisation
list=$(ls)	put ls command in a variable 'list'
nbLines=$((nbLines+1))	increment nbLines
`$0`	the filename of the current script
`$0`	n is a positive decimal number corresponding to the position of an argument (the 1st arg is $1, the 2nd arg is $2, ect)
`$#`	the number of arguments supplied to a script
`$*`	all the arguments are double quoted. If a script receives two arguments, $* is equivalent to $1 $2
`$@`	all the arguments are individually double quoted. If a script receives two arguments, $@ is equivalent to $1 $2
`$?`	the exit status of the last command executed
`$$`	the process number of the current shell. For shell scripts, this is the process ID under which they are executing
`$!`	the process number of the last background command

NOTE: There's no need to specify whether var is string or numerical

---

```
nbLigne=1
list=$(ls)
for i in $list
do
<tab>echo "$nbLigne -> $i"
<tab>nbLigne=$((nbLigne+1))
done
```

Directory commands

`touch fic{1,2}`	creates files: fic1 and fic2
`mkdir folder`	create a folder named 'folder'
`mkdir -p fold1/fold2/fold3`	fold1 contains fold2 and fold2 contains fold3
`mkdir -p fold1/fold2/fold3 toto/tutu`	same as above create fold1 and toto with their sub directories
rm -R *	removes all folders and their subfolders
rm filename	remove a file
rm *.jpg	removes all jpg files

LOOP examples

```
------------WHILE LOOP---------
a=0
while [ $a -lt 10 ]
do
   echo $a
   a=expr $a + 1
done
------------FOR LOOP(1)----------
--
for var in 0 1 2 3 4 5 6 7 8 9
do
   echo $var
done
-----------FOR LOOP(2)-----------
--
for FILE in $HOME/.bash*
do
   echo $FILE
done
-----------FOR LOOP(3)-----------
-
nbLigne=1
for i in $(ls)
do
echo "$nbLigne -> $i"
nbLigne=$((nbLigne+1))
```

LOOP examples (cont)

```
done
----------FOR LOOP(4)-------------
-
for TOKEN in $*
do
   echo $TOKEN
done
----------UNTIL LOOP--------------
--
a=0
until [ ! $a -lt 10 ]
do
   echo $a
   a=expr $a + 1
done
---------SELECT LOOP------------
select DRINK in tea cofee water
juice appe all none
do
   case $DRINK in
      tea|cofee|water|all)
         echo "Go to canteen"
         ;;
      juice|appe)
         echo "Available at home"
      ;;
      none)
         break
      ;;
      *) echo "ERROR: Invalid
selection"
      ;;
   esac
done
-------------SIMPLE BREAK---------
-----
a=0
while [ $a -lt 10 ]
do
   echo $a
   if [ $a -eq 5 ]
   then
```

LOOP examples (cont)

```
      break
   fi
   a=expr $a + 1
done
-----BREAK WITH ARGUMENT----
for var1 in 1 2 3
do
   for var2 in 0 5
   do
      if [ $var1 -eq 2 -a $var2 -
eq 0 ]
      then
         break 2
      else
         echo "$var1 $var2"
      fi
   done
done
```

NOTE: a break command with the argument 2->break out of outer loop and ultimately from inner loop as well.

```
---------CONTINUE-----------
NUMS="1 2 3 4 5 6 7"
for NUM in $NUMS
do
   Q=expr $NUM % 2
   if [ $Q -eq 0 ]
   then
      echo "Number is an even
number!!"
      continue
   fi
   echo "Found odd number"
done
```

Pipes and filters

`grep pattern file(s)print all lines that do not match pattern`

`grep -v`	print all lines that do not match pattern

Pipes and filters (cont)

`grep -n`	print the matched line and its line number
`grep -l`	print only the names of files with matching lines (letter "l")
`grep -c`	print only the count of matching lines
`grep -i`	match either upper- or lowercase
`ls -l | grep -i "carol. *aug"`	find lines with "carol", followed by zero or more other characters abbreviated in a regular expression as ".*"), then followed by "Aug"
`sort fileName`	arranges lines of text alphabetically or numerically
`sort -n`	sort numerically (example: 10 will sort after 2), ignore blanks and tabs
`sort -r`	reverse the order of sort
`sort -f`	sort upper- and lowercase together
`sort +x`	ignore first x fields when sorting

Pipes and filters (cont)

`ls -l | grep "Aug" | sort +4n`	sorts all files in your directory modified in August by order of size, +4n skips four fields (fields are separated by blanks) then sorts the lines in numeric order
`ls -l $dir |egrep "^-|l" |rev |cut -d' ' -f 1|rev`	> list contents starting by '-' or 'l' (file) >reverse letters >cut with delimiter ' '(space) >select field 1 >reverse
`ls -l $dir |egrep "^d" |rev |cut -d' ' -f 1|rev`	> list contents starting by 'd' (directory) >rest same as above

Conditional structure

```
/------------IF_ELIF_FI-----------
-/
#!/bin/sh
a=10
b=20
if [ $a == $b ]
then
   echo "a is equal to b"
elif [ $a -gt $b ]
then
   echo "a is greater than b"
elif [ $a -lt $b ]
then
   echo "a is less than b"
else
   echo "None of the condition met"
fi
```

Conditional structure (cont)

```
RESULT: a is less than b
/-----SIMPLE CASE...ESAC EXAMPLE-----/
FRUIT="kiwi"
case "$FRUIT" in
   "apple") echo "Apple pie is quite tasty."
   ;;
   "banana") echo "I like banana nut bread."
   ;;
   "kiwi") echo "New Zealand is famous for kiwi."
   ;;
esac
RESULT: New Zealand is famous for kiwi.
/----COMPLEXE CASE_ESAC-----/
option="${1}"
case ${option} in
   -f) FILE="${2}"
      echo "File name is $FILE"
      ;;
   -d) DIR="${2}"
      echo "Dir name is $DIR"
      ;;
   *)
      echo "basename ${0}:usage: [-f file] | [-d directory]"
      exit 1 # Command to come out of the program with status 1
      ;;
esac
EXAMPLE RUN OF THE PROGRAME:
$./test.sh
test.sh: usage: [ -f filename ] | [ -d directory ]
$ ./test.sh -f index.htm
$ vi test.sh
$ ./test.sh -f index.htm
File name is index.htm
$ ./test.sh -d unix
Dir name is unix
```

Operators

+, -, *, /, %	Basic arithetic operators
=	Assignment - Assign right operand in left operand
==	Equality - Compares two numbers, if both are same then returns true.
!=	Not Equality - Compares two numbers, if both are different then returns true.
-eq	Checks if the value of two operands are equal or not, if yes then condition becomes true.
-ne	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.
-gt	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true
-lt	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true
-ge	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
-le	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true
!	This is logical negation. This inverts a true condition into false and vice versa

Operators (cont)

-o	This is logical OR. If one of the operands is true then condition would be true
-a	This is logical AND. If both the operands are true then condition would be true otherwise it would be false
-e	check if right operand exists

Input, Output to Promt screen

`read varName`	Store user input in varName
`echo $varName`	Outputs to screen content of $varName
`echo "You entered $varName"`	Same as above

File systems

`who > users`	Puts output of command 'who' in the file 'users' (NOTE: if file already contains content, it will be overwritten)
`cat users`	lists content of file 'users'
`echo new line >> users`	append to last line of file 'users'
`wc -l < users`	get contents of file 'users' as standar input
`command << delimiter document delimiter`	a here document is used to redirect input into an interactive shell script or program
`command > /dev/null`	discard command output
`command > /dev/null 2>&1`	same as above but doesn't display errors. 2 represents STDERR and 1 represents STDOUT.

File systems (cont)

`echo message 1>&2`	display a message on to STDERR by redirecting STDOUT into STDERR

Permissions

```
chmod o+wx,u-x,g=rx testfile
ls -l testfile
-rw-r-xrwx 1 amrood users 1024 Nov 2 00:10 testfile
```

`chown userName filelist`	change ownership of filelist to userName

Navigating file system

`cat filename`	displays contents of filename
`cd dirname`	moves you to dirname directory
`cp file1 file2`	copies 1 file/directiry to specified location
`file filename`	identifies the fie type(binary, tect, etc..)
`find filename dir`	finds a file/directory
`head filename`	shows the begining of a file
`less filename`	browses through a file from begining to end
`ls dirname`	shows contents of directory
`mkdir dirname`	creates speicified directory
`more filename`	browses through a file from begining to end
`mv file1 file2`	moves the location of or renames a file/directory

Navigating file system (cont)

`pwd`	shows the current directory the users is in
`rmdir dirname`	removes a a directory
`rm filename`	removes a file
`tail filename`	shows the end of a file
`touch filename`	creates a blank file or modifies an existing file's attrbites
`whereis filename`	shows the location of a file
`which filename`	shows the location of a file if it is in your path
`df -k`	displays disk space sage in kilobytes
`du dirname` or `du -h dirname`	show disk usage on particular directory
`mount`	view what is currently mounted
`mount -t fileSysType deviceToMount dir_to_mount_to`	mount a filesystem (CD etc..)
`mount -t iso9660 /dev/cdrom /mnt/cdrom`	example of above command
`unmount mountPointOrDevice`	unmount a filesystem
`quota`	displays disk usage and limits for a user of group

```
echo $(find $dir -type f -printf "%f\n") |tr " " "\n"
```

What this does?

> display path of contents of dir
> select only content of type file -f
> print the result in the same line
> tr (translate) ' ' (space) into '\n' line break