Bash Scripting Language Cheat Sheet Cheat Sheet

by danilobanjac via cheatography.com/50162/cs/13858/

Commands Line Commands	
clear	Clears the screeen.
vim filename.sh	Creates a .sh file.
sh filename.sh	Execute Bash Script.
./filename.sh	Other way of executing bash script.
ls -l	See all file permissions inside a folder.
chmod +x filename.sh	This will add execute permission to the file.
ls --help	This will open all available commands.
ls --help | grep "\-U"	This will grab more information about "-U" command.
touch wood.txt	This will create a file.
echo "here is something use it" > wood.txt	This will write to the file that we first created.
cat wood.txt	This will get us the output of the file.
cat {testfile01,testfile02} > test00	This will take as many files as we want and store the content of those file in one file in this case "test00".
echo "here is something new" > wood.txt	This will replace the whole content of the wood.txt file with this new content, if we want to add to file we need to use the command from below.

Commands Line Commands (cont)	
echo "here is comething new" >> wood.txt	Double >> will add to file.
: > wood.txt	This will empty the whole file, remeber ":" that says do nothing.
rm wood.txt	Will remove the file completely.
touch test1 test2 test3 test4	for example if we create a hundreds of file with similar name how we could delete them all or select them all? (Solution below).
rm test*	("*" astrix sign will mark everything that begins with test and delete it.

Vim Commands	
Press "a"	In order to start editing the file.
Press "esc"	In order to quit editing the file.
Write ":"	To allow you to save or quit or write to file.
Write ":wg"	To write those changes and quit with saving.
Write ":wg!"	To force this action.
Write ":q!"	To quit without saving.

By **danilobanjac**
cheatography.com/danilobanjac/

Not published yet.
Last updated 17th December, 2017.

Part One (Variables and Parameters)

```
ECHOING VARIABLES:
- var=10
echo var (This will echo "var")
echo $var (This will echo the actual value, this is
why we are using dollar sign "$" before variable. So
that program knows that we want a value from the
variable)
UNSET:
- var=10
unset var
echo $var ("unset" will actually reassign the value of
"var" to null)
ASIGN VALUE TO THE VARIABLE THROUGH USER:
- echo "type in some value"
read var2 (This will ask user for input same like
"prompt" in javascript and store this value to
variable)
echo $var2
PROPERTIES OF VARIABLES:
- var="T r a l a l a lalalal l"
echo $var (This will output the string but we will
miss some spaces, this means that ot all of our spaces
will be printed that is why we are using "")
var="T r a l a l a lalalal l"
echo "$var" (When we use "" to wrap our variable with
as well the "$" dollar sign then all of our spaces
will be outputed. This is recommended way of doing
"echo" in bash)
DEFINE NULL VARIABLE:
- var= (This will define variable with a value of
"null")
DECLARING VARIABEL ON SAME LINE:
- var1=11 var2=22 var3=33
echo "$var1 $var2 $var3" (Same like in javascript)
ASSIGNING, REASSIGNING AND UNSETTING THE VALUE:
- var=
echo "$var"
```

Part One (Variables and Parameters) (cont)

```
var=9
echo "$var"
var=10
echo "$var"
unset var
echo "$var"
ASSIGN VALUE FROM BASH COMMAND TO VARIABLE:
- hi=$(ls -la) (This is the way of doing it)
          echo "$hi" (Will print value of "hi")
ADDING TO A VARIABLE OR DOING ARITHMETIC OPERATIONS ON
VARIABLES:
- var=
let "var ++"
echo "$var" (This will output 1)
let "var += 10"
echo "$var" (This will output 10)

REPLACING VALUES IN NUMBER:
- num=1100
var=${num/10/B}
echo "$var" (This will echo "1B0")
ENVIRONMENTAL VARIABLES:
        - #!/bin/bash
          MIN=10 (Minimum passed arguments in order to
execute the script)
          if [ -n "$1" ]; then echo "1st one is $1";
fi ("$1" first argument that is being passed to script
and "-n" check if argument exists. Returning either
"true" or "false")
          if [ -n "$2" ]; then echo "2st one is $2";
fi
          if [ -n "$3" ]; then echo "3rd one is $3";
fi
          if [ -n "$4" ]; then echo "4th one is $4";
fi
          if [ -n "$5" ]; then echo "5th one is $5";
fi
          if [ -n "$6" ]; then echo "6th one is $6";
fi
```

Part One (Variables and Parameters) (cont)

```
          if [ -n "$7" ]; then echo "7th one is $7";
fi
          if [ -n "$8" ]; then echo "8th one is $8";
fi
          if [ -n "$9" ]; then echo "9th one is $9";
fi
          if [ -n "${10}" ]; then echo "10th one is
${10}"; fi
         echo "List of arguments: "$ "" ("$" this will
take all the arguments that are being passed to script)
         echo "Name of Script: \""$0"\"" ("$0" is used
to grab the name of the file)
         if [ $# -lt "$MIN" ]; then echo "Not enought
arguments, need "$MIN" to run!"; fi (This check "$#"
number of all arguments being passed to script and
compare it to our defined variable.)
         - sh filename.sh 1 2 3 4 5 6 7 8 9 10
(Passing arguments to our script and printing them)
```

Part Two (Return Values)

```
RETURN VALUES:
         - #!/bin/bash
           NO_OF_ARGS=2
           E_BADARGS=85
           E_UNREADABLE=86
           if [ $# -ne "$NO_OF_ARGS" ]; then echo
"Usage: "$0" fileOne fileTwo"; exit $E_BADARGS; fi
           if [ ! -r "$1" ] || [ ! -r "$2" ]; then
echo "One or both files does not exist: "$1" or
"$2""; exit "$E_UNREADABLE"; fi
           (
           cmp $1 $2
           ) > /dev/null 2>&1
           if [ $? -eq 0 ]; then echo "Files are the
same!"; else echo "Files are not the same!"; fi
           exit 0
         - Explanation: This script will accept two
files and than compare if they are identical. We are
defining our "EXIT CODES" and returning them depending
on situation. (exit code 0 == good (true); exit code 1
== bad (false))
```

Bash Script Programming Language Basics

```
ECHO:
- echo "Some text" (This is same like "print" in
python. It will print to command line)
- echo "Some text" #This is comment ("This is
comment" will not be executed)
DEFINING VARIABLES:
- name=10 (This desfines a variable with integer)
- name=tea (This defines a variable)
USING HASH "#":
- echo "The word $name contains ${#name} chars" (This
is how to do a string formatting in bash)
- $(#name) (This will return a lenght of the string in
bash)
- $name will replace this part of the string with the
current variable.
USING SEMICOLON:
- echo "hi there"; echo "you there?" (; semicolon sign
will tell bash to run this as a next line of the code)
IF/THEN/ELSE STATEMENT:
- var=10
- if [ "$var" -gt 5 ]; then echo "YES"; else echo
"NO"
fi (Please note the spaces inside the brackets they
need to be there for statement to run, and every
semicolon will be seen as a next line and at the end we
have "fi" this closes the if statement)
FOR LOOP:
- colors="red black white" (This defines a variable as
a string)
- for col in $colors
do (This will run the desired action)
echo "$col" (This will echo "col")
done (This will finish the for loop)
- if we run this script now the result below will be
printed to console:
red
black
white
```

Bash Script Programming Language Basics (cont)

- as you can see for loop threated variable colors as a list. Because "for col in $colors" dollar sign and colors will make a list from the string. If we put $colors inside double quotes like this "$colors" this will be then threated as a string and the ouput to the console will be:
red black white
- if we put single quotes '$colors' this would take the actual word $colors and print it.
USING LET:
- let "y=((x=20, 10/2))" (let in bash will let us to perform arithemic operations on variables)
- echo $y (This will return 5 because we separated the operation with comma)
CHANGING THE STRING TO UPPER OR LOWER:
- var = DSLConnection
- echo ${var,} (This will change the first character of the string to lower)
- echo ${var,,} (This will change the whole string to lower)
USING "\" ESCAPE CHARACTER:
- echo ""Linux is awesome""
will output this Linux is awesome to the console.
- echo "\"Linux is awesome"\" (This will take quotes literaly)
will output this "Linux is awesome" to the console
REASIGN THE VALUE:
- let val=500/2
val2=echo $val (This will allow us to reasign the value from the first variabel to the second variabel)
echo "$val2"
will give same output "250"
IMPORTANT
In order to get the value from any oder script example Python. Bash script will only recognise the value if value is printed, that means that function can return number but at the end when we call the function in python script it should look like this:
print return_value()
USING ":" SIGN:

Bash Script Programming Language Basics (cont)

- var=20
if ["$var" -gt 15]; then :;else echo "$var";fi (This ":" sign after then will actually tell our code to do nothing this will come very useful, same like "pass" in python)
IF STATEMENT USING "?" MARK:
- var=10
echo $((var2=var1<20?1:0)) (This will return the first value "1" if statement is true and second value if statement is false. In this case we don't need the "$" sign before "var1" to tell the program to use value, this is special case)
CREATE ARRAY:
- Colors=(red blue green white) (This will create an array in bash)
WRAP STRINGS INTO SOME CHARACTERS:
- echo \+{test1,test2,test3}\+ (You can replace the "+" sign with any other sing for example "$")
will output +test1+ +test2+ +test3+ to console
CREATE RANGE:
- echo {0..9} (This will print all numbers between 0 and 9 (including 9) same like range() in python)
SEPARATE THE BLOCK OF CODE:
- var1=1
var2=2
{
var1=10
var2=12
}
echo "$var1 $var2" (The output to the console will be 10 and 12 because "{}" will separate this part of code)
SAVE EXIT CODE FROM THE LAST COMMAND:
- python myPythonScript.py
ret=$?
if [$ret -ne 0]; then

Bash Script Programming Language Basics (cont)

```
#Handle failure
#exit if required
fi
USE EXIT CODE TO MANIPULATE SCRIPT:
- #!/bin/bash
touch /root/test 2> /dev/null
if [ $? -eq 0 ]
then
echo "Successfully created file"
exit 0
else
echo "Could not create file" >&2
exit 1
fi
HIDE WHOLE OUTPUT FROM THE SCRIPT:
- (
    ./manage.py create_test_database
) > /dev/null 2>&1
CONNECTING IF STATEMENTS:
- var=1
if [ "$var" -gt 0 ] && [ "$var" -eq 10 ]; then echo
"THEN PART"; else echo "HELLOOO"; fi (Example with
logical "and" statment)
var=1
if [ "$var" -gt 0 ] || [ "$var" -eq 10 ]; then echo
"THEN PART"; else echo "HELLOOO"; fi (Example with
logical "or" statment)
MODULO:
  - let var=5%4
echo "$var" (Result will be one)
STRING UPPER CASE:
```

Bash Script Programming Language Basics (cont)

```
- some_word=tEsT
echo "${some_word^}" (This will grab first letter and
make it upper case)
echo "${some_word}" (This will grab whole word and
make it upper case)
```