

Bash Script

Cheat Sheet



	Basic Syntax		Variables
#! /bin/bash	Shebang at the beginning of a script specifies the interpreter	var_name=val	Assign a value to the specified variable
#! /usr/bin/env bash	Alternative shebang -using environment variable	\$ var_name	Access the value of the specified variable
\$#	Stores the number of argument passes to the Bash script	"\$var_name"	Variables with special bash script character at the beginning must be quoted with double quotes or single quotes
\$1,\$2,\$3	Variables that store the values passed as arguments to the Bash script	var_name=\$(co mmand)	Assign the output of a command to the specified variable
exit	Exit from the Bash script	readonly var_name=val	Prevent the value of a specified variable to be modified
CTRL + C	Keyboard shortcut to stop Bash	\$HOME, \$PATH, \$USER etc.	Few predefined environment variables
\$ (command)	Execute a command inside a subshell	\$0	Predefined varibles that stores the name of the script
sleep	Pause for a specified number of seconds, minutes, hours or days	\$#	Predefined variables that stores the number of command line arguments
		#?	Predefined variable that stores the exit status of the last executed command
	Comments	\$\$	Predefined variable that stores the process ID of the current script
#	Single line comment. The text comes after it will not be executed	\$!	Predefined variable that stores the proces ID of the last background command
: <<' '	Multiple line comment	unset var_name	Delete a variable with specified name

Command Execution		Input/Output	
command_nam e	Directly execute the command with specified name	read -p	Prompt the user for information to enter
`variable_name =command``	Older version of substituting the output of the command to a specified variable	command < input_file	Redirect input from a file to a command
command > file_name	Redirect the output of a command to a specified file	command 2> error_file	Redirect standard error to a specified file
command >> file_name	Redirect the output of a command to a specified command and append it with the existing content	command &> file_name	Redirect standard output and standard error to a specified file
command1 command2	Use the standard output of command1 as the standard input of command2		



Prepared By:

Md Zahidul Islam Laku

Bash Script

Cheat Sheet



Copyright ©2023 linuxsimply.com | All rights reserved.

	Loops		Conditional Statements
for variable in list; do # Code done	Iterate over the list and execute code for each element of the list	if [condition]; then #code fi	Test a condition and execute the then clause if it is true
while condition; do # Code done	Execute code repeatedly as long as the condition is true	<pre>if [condition]; then #code fi else #code fi</pre>	Execute the then clause if the condition is true, otherwise execute the else clause
until condition; do # Code done	Execute code repeatedly until the condition becomes true	if [condition1]; then #code elif [condition2]; then #code else #code fi	Execute the then clause if the condition is true or execute the elif clause if the condition is true, otherwise execute the else clause
select variable in list; do # Code done	Execute code based on the choice that the variable takes from the list	case variable in pattern1) #code ;; pattern2) #code ;; pattern3) #code ;; *) ;; esac	Execute code following each pattern if the variable matches the pattern otherwise execute * if none of the patterns match
continue	Skip the current iteration of a loop and continue with the next iteration	test condition	Returns 0 or 1 indicating whether the condition is true or false
break	Terminate a loop based on certain condition		
			Arithmetic Operations
	Data Types	+	Addition
x=5	Integer or floating point values are treated as Number	-	Subtraction



Bash Script

Cheat Sheet



	Data Types		Arithmetic Operations
is_valid=0	Boolean value represent False	*	Multiplication
is_valid=1	Boolean value represents True	/	Division
declare -a var	Declare an indexed array	%	Modulus or remainder
declare -A var	Declare an associated array	**	Raise to a power
declare -i var	Declare an integer variable	((i++))	Increment a variable
declare -r var	Declare a read only variable	((i))	Decrement a variable
declare -x var	Declare an exported variable		
var_name=""	Absence of value or uninitialized variable		Function
array=("elemen t1" "element2" "element3")	A collection of elements accessed using numerical indices	<pre>function_name() { # code }</pre>	Declare a function with specified function name
declare -A array1 array1["elemen t1"]="value1" array2["elemen t2"]="value2"	A collection of elements accessed using string indices	function_name	Call a function with specified function name
var="Hellow World"	Sequence of characters enclosed in single or double quotes is treated as String	local var_name=val	Declare a local variable inside a function
		return	Exit a function and return a value of the calling function
	Baralana Garantana		

	Boolean Operators	
&&	Logical AND operator	
П	Logical OR operator	
!	NOT equal to operator	

	String Comaprison Opearators
=	equal
!=	not equal
<	less then
>	greater then
-n str1	string str1 is not empty
-z str2	string str2 is empty

	Arithmetic Conditional Operators
-lt	Equals to mathematical < operator(less than)
-gt	Equals to mathematical > operator(greater than)
-le	Equals to mathematical <= operator(less than equal)
-ge	Equals to mathematical >= operator(greater than equal)
-eq	Equals to mathematical == operator(equal)
-ne	Equals to mathematical != operator(not equal)



Bash Script

Cheat Sheet



String Manipulation		
concatenated=" \$str1 \$str2"	Concatenate the variables set in str1 and str2	
substring=\${str: n}	Extracts a substring from n-th index to till the end of the string that stored in variable str	
substring=\${str: 0:5}	Extracts substring from 0-th index to 5-th index of the string that stored in variable str	
length=\${#str}	Find the length of the string that stored in variable str	
[[\$str == *"World"*]]	Returns True if the string stored in variable str contains the word World	
•	Replaces the first occurrence of 'World' with 'Universe' within the string stored in str variable	
-	Trims leading whitespace of the string	
trimmed=\${trim	Trims trailing whitespaces of the string	

stored in trimmed variable

Prepared By:

med%%*()}

Md Zahidul Islam Laku

Copyright ©2023 linuxsimply.com | All rights reserved.