Syntax/Command/Variable	Desc	Example
$0	The name of the Bash script.	echo $0
$1 - $9	The first 9 arguments to the Bash script. (As mentioned above.)	echo $1 $2 $3
$#	How many arguments were passed to the Bash script.	echo $#
$@	All the arguments supplied to the Bash script.	echo $@
$?	The exit status of the most recently run process.	echo $?
$$	The process ID of the current script.	echo $$
$USER	The username of the user running the script.	echo $USER
$HOSTNAME	The hostname of the machine the script is running on.	echo $HOSTNAME
$SECONDS	The number of seconds since the script was started.	echo $SECONDS
$RANDOM	Returns a different random number each time is it referred to.	echo $RANDOM
$LINENO	Returns the current line number in the Bash script.	echo $LINENO
x=value	Set **x** to **value**	x=123 x=hello x='Hello World' x="Hello World"
$x	Read the value of **x**	echo $x
x=$(command)	Set **x** to output of a command	x=$(echo 123)
export x	Export **x** to all child process	export x
read x	Read input and write it in **x**	read x read -p "User: " x read -sp "Pass: " x
let <arithmetic expression>	To do simple arithmetic	let a=5+4 let "a = 5 + 4" let a++
expr item1 operator item2	To do simple arithmetic (print the result)	expr 11 % 2 a=$(expr 10 - 3)
$((expression))	to do basic arithmetic	a=$((4 + 5)) b=$((a + 3)) $((b += 3))
${#x}	find out the length of a variable (**x**)	a='Hello World' echo ${#a}
if [<some test>] **then** **<commands>** **fi**	If statement in Bash	if [$1 -gt 100] then echo Hey that\'s a large number. pwd fi

test/[expr]	To test or evaluate expression	! EXPRESSION -n STRING -z STRING STRING1 = STRING2 STRING1 != STRING2 INTEGER1 -eq INTEGER2 INTEGER1 -gt INTEGER2 INTEGER1 -lt INTEGER2 -d FILE -e FILE -r FILE -s FILE -w FILE -x FILE
if [<some test>] **then** **<commands>** **elif [<some test>]** **then** **<different commands>** **else** **<other commands>** **fi**	IF/Else IF/Else in Bash	if [$1 -ge 18] then echo You may go to the party. elif [$2 == 'yes'] then echo You may go to the party but be back before midnight. else echo You may not go to the party. fi
and - && **or - ||**	And / Or Boolean operator in Bash	if [-r $1] && [-s $1] then echo This file is useful. fi
case <variable> in **<pattern 1>)** **<commands>** **;;** **<pattern 2>)** **<other commands>** **;;** **esac**	Case / Switch statement in Bash	case $1 in start) echo starting ;; stop) echo stoping ;; restart) echo restarting ;; *) echo don\'t know ;; esac
while [<some test>] **do** **<commands>** **done**	While loop in Bash	counter=1 while [$counter -le 10] do echo $counter ((counter++)) done

until [<some test>] **do** **<commands>** **done**	Until loop in Bash	counter=1 until [$counter -gt 10] do echo $counter ((counter++)) done
for var in <list> **do** **<commands>** **done**	For loop in Bash	names='Stan Kyle Cartman' for name in $names do echo $name done
{1..5}	Ranges in Bash	for value in {1..5} do echo $value done for value in {10..0..2} do echo $value done
select var in <list> **do** **<commands>** **done**	Select from options/menu in Bash	names='Kyle Cartman Stan Quit' PS3='Select character: ' select name in $names do if [$name == 'Quit'] then break fi echo Hello $name done

function_name () {
<commands>
}
	Define a function in Bash	print_something () {
		echo Hello I am a function
		}
		print_something
		
		print_something2 () {
		echo Hello $1
		}
		print_something2 Mars

local x=value	Local variable in Bash	print_something () {
		local x=123
		echo $x
		}
		print_something

name[index]=value **declare -a name=('value1 'value2');**	Define an array	Unix[0]='Debian' declare -a Unix=('Debian' 'Red hat' 'Red hat' 'Suse' 'Fedora');

${Array[@]}	Print all array	declare -a Unix=('Debian' 'Red hat' 'Red hat' 'Suse' 'Fedora'); echo ${Unix[@]}
#array[@]	Get array length	declare -a Unix=('Debian' 'Red hat' 'Suse' 'Fedora'); echo ${#Unix[@]}
${#array[I]}	Length of the nth Element in an Array	declare -a Unix=('Debian' 'Red hat' 'Suse' 'Fedora'); echo ${#Unix[1]}
${array[@]:X:Y}	Extraction by offset and length for an array	Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora' 'UTS' 'OpenLinux'); echo ${Unix[@]:3:2}
${array[I]:X:Y}	Extraction with offset and length, for a particular element of an array	Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora' 'UTS' 'OpenLinux'); echo ${Unix[2]:0:4}
${array[@]/find/replace}	Search and Replace in an array elements	Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora' 'UTS' 'OpenLinux'); echo ${Unix[@]/Ubuntu/SCO Unix}
unset array[i]	Remove an Element from an Array	Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora' 'UTS' 'OpenLinux'); echo ${Unix[3]}
newarray=("${array[@]}")	Copying an Array	Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora' 'UTS' 'OpenLinux'); Linux=("${Unix[@]}") echo ${Linux[@]}
Array1=('value1') **Array2=('value2')**	Concatenation of two Bash Arrays	Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora' 'UTS' 'OpenLinux'); Shell=('bash' 'csh' 'jsh' 'rsh' 'ksh' 'rc' 'tcsh'); UnixShell=("${Unix[@]}" "${Shell[@]}") echo ${UnixShell[@]} echo ${#UnixShell[@]}