|  |  |
| --- | --- |
|  | **Basic Syntax** |
| #! /bin/bash | Shebang at the beginning of a scriptspecifies the interpreter |
| #! /usr/bin/envbash | Alternative shebang -using environmentvariable |
| $# | Stores the number of argument passes to the Bash script |
| $1 , $2, $3 | Variables that store the values passed asarguments to the Bash script |
| exit | Exit from the Bash script |
| CTRL + C | Keyboard shortcut to stop Bash |
| $ (command) | Execute a command inside a subshell |
| sleep | Pause for a specified number of seconds,minutes, hours or days |

|  |  |
| --- | --- |
|  | **Variables** |
| var\_name=val | Assign a value to the specified variable |
| $ var\_name | Access the value of the specified variable |
| “$var\_name” | Variables with special bash script character at the beginning must be quoted with double quotes or singlequotes |
| var\_name=$(command) | Assign the output of a command to thespecified variable |
| readonlyvar\_name=val | Prevent the value of a specified variableto be modified |
| $HOME, $PATH,$USER etc. | Few predefined environment variables |
| $0 | Predefined varibles that stores the nameof the script |
| $# | Predefined variables that stores thenumber of command line arguments |
| #? | Predefined variable that stores the exitstatus of the last executed command |
| $$ | Predefined variable that stores theprocess ID of the current script |
| $! | Predefined variable that stores the proces ID of the last backgroundcommand |
| unset var\_name | Delete a variable with specified name |

|  |  |
| --- | --- |
|  | **Comments** |
| # | Single line comment. The text comes after it will not be executed |
| : <<' ' | Multiple line comment |

|  |  |
| --- | --- |
|  | **Command Execution** |
| command\_name | Directly execute the command withspecified name |
| `variable\_name=command`` | Older version of substituting the outputof the command to a specified variable |
| command >file\_name | Redirect the output of a command to aspecified file |
| command >> file\_name | Redirect the output of a command to aspecified command and append it with the existing content |
| command1 |command2 | Use the standard output of command1as the standard input of command2 |

|  |  |
| --- | --- |
|  | **Input/Output** |
| read -p | Prompt the user for information to enter |
| command <input\_file | Redirect input from a file to a command |
| command 2>error\_file | Redirect standard error to a specified file |
| command &> file\_name | Redirect standard output and standard error to a specified file |

*Prepared By:* [*Md Zahidul Islam Laku*](https://linuxsimply.com/md-zahidul-islam-laku/) *Copyright ©2023 linuxsimply.com| All rights reserved.*

|  |  |
| --- | --- |
|  | **Loops** |
| for variable in list; do# Codedone | Iterate over the list and execute code for each element of the list |
| while condition; do# Code done | Execute code repeatedly as long as the condition is true |
| until condition; do# Code done | Execute code repeatedly until the condition becomes true |
| select variable in list; do# Code done | Execute code based on the choice that the variable takes from the list |
| continue | Skip the current iteration of a loop andcontinue with the next iteration |
| break | Terminate a loop based on certaincondition |

|  |  |
| --- | --- |
|  | **Conditional Statements** |
| if [ condition ]; then#codefi | Test a condition and execute the then clause if it is true |
| if [ condition ]; then#code fielse#code fi | Execute the then clause if the condition is true, otherwise execute the else clause |
| if [ condition1 ]; then#codeelif [ 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 |
| 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 |
| test condition | Returns 0 or 1 indicating whether thecondition is true or false |

|  |
| --- |
| **Data Types** |
| x=5 | Integer or floating point values aretreated as Number |

|  |  |
| --- | --- |
|  | **Arithmetic Operations** |
| + | Addition |
| - | Subtraction |

|  |  |
| --- | --- |
|  | **Data Types** |
| is\_valid=0 | Boolean value represent False |
| is\_valid=1 | Boolean value represents True |
| declare -a var | Declare an indexed array |
|  declare -A var  | Declare an associated array  |
| declare -i var | Declare an integer variable |
|  declare -r var  | Declare a read only variable  |
| declare -x var | Declare an exported variable |
| var\_name="" | Absence of value or uninitialized variable |
| array=("elemen t1" "element2" "element3"...) | A collection of elements accessed using numerical indices |
| declare -A array1 array1["elemen t1"]="value1" array2["element2"]="value2" | A collection of elements accessed using string indices |
| var="Hellow World" | Sequence of characters enclosed in single or double quotes is treated asString |

|  |  |
| --- | --- |
|  | **Arithmetic Operations** |
| \* | Multiplication |
| / | Division |
| % | Modulus or remainder |
| \*\* | Raise to a power |
|  ((i++))  | Increment a variable  |
| ((i--)) | Decrement a variable |

|  |  |
| --- | --- |
|  | **Function** |
| function\_name() {# code} | Declare a function with specified function name |
| function\_name | Call a function with specified function name |
| local var\_name=val | Declare a local variable inside a function |
| return | Exit a function and return a value of thecalling function |

|  |  |
| --- | --- |
|  | **Boolean Operators** |
| && | Logical AND operator |
| || | Logical OR operator |
| ! | NOT equal to operator |

|  |  |
| --- | --- |
|  | **Arithmetic Conditional Operators** |
| -lt | Equals to mathematical < operator(lessthan) |
| -gt | Equals to mathematical >operator(greater than) |
| -le | Equals to mathematical <= operator(lessthan equal) |
| -ge | Equals to mathematical >=operator(greater than equal) |
| -eq | Equals to mathematical ==operator(equal) |
| -ne | Equals to mathematical != operator(notequal) |

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

|  |  |
| --- | --- |
|  | **String Manipulation** |
| concatenated="$str1 $str2" | Concatenate the variables set in str1 andstr2 |
| substring=${str: n} | Extracts a substring from n-th index to till the end of the string that stored invariable str |
| substring=${str: 0:5} | Extracts substring from 0-th index to 5-th index of the string that stored in variablestr |
| length=${#str} | Find the length of the string that storedin variable str |
| [[ $str ==\*"World"\* ]] | Returns True if the string stored invariable str contains the word World |
| replaced=${str/ World/Universe} | Replaces the first occurrence of 'World'with 'Universe' within the string stored in str variable |
|  trimmed=${str#  | Trims leading whitespace of the string  |
| trimmed=${trimmed%%\*( )} | Trims trailing whitespaces of the stringstored in trimmed variable |

*Prepared By:* [*Md Zahidul Islam Laku*](https://linuxsimply.com/md-zahidul-islam-laku/) *Copyright ©2023 linuxsimply.com| All rights reserved.*