### Operators...

### 1. ****Arithmetic Operators****

Used to perform basic arithmetic calculations.

| Operator | Description | Example |
| --- | --- | --- |
| + | Addition | echo $((5 + 3)) |
| - | Subtraction | echo $((5 - 3)) |
| \* | Multiplication | echo $((5 \* 3)) |
| / | Division | echo $((6 / 3)) |
| % | Modulus (remainder) | echo $((5 % 2)) |
| \*\* | Exponentiation | echo $((2 \*\* 3)) |

### 2. ****Comparison Operators (Arithmetic)****

Used to compare numbers in conditional expressions.

| Operator | Description | Example |
| --- | --- | --- |
| -eq | Equal to | [ 5 -eq 5 ] |
| -ne | Not equal to | [ 5 -ne 4 ] |
| -lt | Less than | [ 3 -lt 5 ] |
| -le | Less than or equal to | [ 5 -le 5 ] |
| -gt | Greater than | [ 5 -gt 3 ] |
| -ge | Greater than or equal to | [ 5 -ge 3 ] |

### 3. ****Comparison Operators (String)****

Used for string comparisons.

| Operator | Description | Example |
| --- | --- | --- |
| = | Equal to | [ "$str1" = "$str2" ] |
| != | Not equal to | [ "$str1" != "$str2" ] |
| < | Less than (lexicographically) | [ "$str1" \< "$str2" ] |
| > | Greater than (lexicographically) | [ "$str1" \> "$str2" ] |
| -z | String is empty | [ -z "$str" ] |
| -n | String is not empty | [ -n "$str" ] |

**Note**: < and > must be escaped (with \) inside single brackets [].

### 4. ****Logical Operators****

Used to combine multiple conditions.

| Operator | Description | Example |
| --- | --- | --- |
| && | Logical AND | [[ $num -gt 0 && $num -lt 10 ]] |
| ` |  | ` |
| ! | Logical NOT | [[ ! $num -eq 5 ]] (True if $num is not 5) |

### 5. ****File Test Operators****

Used to test properties of files and directories.

| Operator | Description | Example |
| --- | --- | --- |
| -e | File exists | [ -e file.txt ] |
| -f | File is a regular file | [ -f file.txt ] |
| -d | File is a directory | [ -d dir\_name ] |
| -r | File is readable | [ -r file.txt ] |
| -w | File is writable | [ -w file.txt ] |
| -x | File is executable | [ -x file.sh ] |
| -s | File has non-zero size | [ -s file.txt ] |
| -L | File is a symbolic link | [ -L symlink.txt ] |
| -p | File is a named pipe (FIFO) | [ -p pipe\_name ] |
| -S | File is a socket | [ -S socket\_file ] |

### 6. ****Bitwise Operators****

These are used for bit-level operations.

| Operator | Description | Example |
| --- | --- | --- |
| & | Bitwise AND | echo $((5 & 3)) |
| ` | ` | Bitwise OR |
| ^ | Bitwise XOR | echo $((5 ^ 3)) |
| ~ | Bitwise NOT | echo $((~5)) |
| << | Left shift | echo $((5 << 1)) |
| >> | Right shift | echo $((5 >> 1)) |

### 7. ****Assignment Operators****

Used for assigning values to variables.

| Operator | Description | Example |
| --- | --- | --- |
| = | Assigns value | var=5 |
| += | Increment and assign | var+=1 |
| -= | Decrement and assign | var-=1 |
| \*= | Multiply and assign | var\*=2 |
| /= | Divide and assign | var/=2 |
| %= | Modulus and assign | var%=2 |

### 8. ****Regular Expression Matching (Bash-specific)****

Used to check if a string matches a regular expression.

| Operator | Description | Example |
| --- | --- | --- |
| =~ | Regex match (Bash only) | [[ $str =~ ^[0-9]+$ ]] |

### 9. ****Redirection Operators****

Used to redirect input/output in Bash.

| Operator | Description | Example |
| --- | --- | --- |
| > | Redirect standard output to file | echo "text" > file.txt |
| >> | Append standard output to file | echo "more text" >> file.txt |
| < | Redirect input from file | cat < file.txt |
| 2> | Redirect standard error | command 2> error.log |
| &> | Redirect both stdout and stderr | command &> output.log |
| ` | ` | Pipe output to another command |

### 10. ****Miscellaneous Operators****

Other useful operators in Bash.

| Operator | Description | Example |
| --- | --- | --- |
| , | Sequential execution | echo "a", echo "b" |
| ; | Command separator | command1; command2 |
| & | Run command in background | command & |
| () | Execute command in subshell | (cd /tmp; ls) |
| {} | Group commands in current shell | { echo "a"; echo "b"; } |