Basic Shell operators

There are many operators supported by each Shell. Our tutorial is based on the default Shell (Bourne) so we are discussing Bourne shell operators in this chapter.

There are many operators supported by each Shell. Our tutorial is based on the default Shell (Bourne) so we are discussing Bourne shell operators in this chapter.

There are the following operators that we need to consider:

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. String operators
5. File check operators

Bourne shell at first did not have any technique to present simple arithmetic, but it used peripheral programs, either **awk** or a simpler program than **expr** .

Here is a simple example to add two numbers:

```
#! / bin / sh val = `expr 2 + 2` echo " Total value: $ val "
```

It will produce the following result:

```
Total value : 4
```

The points you need to keep in mind are:

There must be a space between the two operators and two expressions, for example 2 + 2 is not correct, it should be written as 2 + 2.

Full expression should be surrounded by ``, called an inverse comma.

Arithmetic operators in Unix / Linux

The following arithmetic operators are supported by Bourne shell.

Suppose variable a holds value 10 and variable b holds value 20, then:

For example:

Operator Description Example + Addition - add value on each side to the operator `expr $ a + $ b` the result is 30 - Subtraction - except the right value of the left operator` expr $ a - $ b `result is -10 * Multiplication -

multiplying the value on each side with the operator. `expr $ a * $ b` the result is 200 / Divide - divide the right value for the left value` expr $ b / $ a` the result is 2% Take the balance - take the rest after dividing the left value by the right value `expr $ b% $ a` results 0 = assignment - assigning the left operand to the right operand a = $ b will assign the value of b to a == equals - Comparing two numbers, if both are the same, the result is true. [$ a == $ b] will return False. ! = Unbalanced magic - compare two numbers, if both numbers are different then the value returned is true. [$ a! = $ b] will return True result.

It is very important to remember here that all conditional expressions will be enclosed in square brackets ([]) with a blank space around them, for example [$ a == $ b] is true, and [$ a == $ b] is incorrect.

All arithmetic operations are performed using long integers.

Relational operators in Unix / Linux

Bourne shell supports the following relational operators that are specific to numeric values. These operators do not work for strings unless its value is numeric.

For example, the following operators will check a relationship between 10 and 20, also between "10" and "20" but not between "ten" and "twenty '.

Suppose the variable a holds the value 10 and the variable b holds the value 20, then:

For example

Operator DescriptionExample-eq Checks whether the value of two operands is balanced or not, if any, the condition becomes correct. [$ a -eq $ b] is incorrect. -ne Check whether the values ??of the two operands are balanced or not, if not equal, the condition becomes correct. [$ a -ne $ b] is correct. -gt Check if the value of left operand is greater than the value of the right operand, if true, the condition becomes true. [$ a -gt $ b] is incorrect. -lt Check if the value of the left operand is less than the value of the right operand, if true, the condition becomes true. [$ a -lt $ b] is correct. -ge Check if the value of the left operand is greater than or equal to the value of the right operand, if true, the condition becomes true. [$ a -ge $ b] is incorrect. -le Check if the value of left operand is less than or equal to the value of the right operand, if true, the condition becomes true. [$ a -le $ b] is correct.
It is very important to remember here that all conditional expressions will be enclosed in square brackets ([]) with blank spaces around them, for example [$ a = $ b] is correct and [$ a = $ b] is incorrect.

Logical operators in Unix / Linux

There are the following logical operators supported by Bourne Shell

Assuming the variable a holds the value 10 and the variable b holds the value 20, the following is an example of using all logical operators.

For example

Description Operator For example! Negative. If the condition is true then the value is wrong and vice versa. [! false] is true. -Or spell or. If one of the operands is correct, the condition is true. [$ a -lt 20 -o $ b -gt 100] is true. -a Magic and. If both operands are correct, the condition is true, otherwise false . [$ a -lt 20 -a $ b -gt 100] is false.

String operators in Unix / Linux

The following string operators are supported by Bourne Shell.

Suppose variable a holds the value "abc" and variable b holds the value "efg":

For example

Operator DescriptionExample = Check if the value of two operands is balanced or not, if yes, the condition is true. [$ a = $ b] is not correct. ! = Check if the value of the two operands is equal or not, if not equal, the condition is correct. [$ a! = $ b] is correct. -z Check if the string operand size is set to 0. If it is 0, it returns true. [-z $ a] is incorrect. -n Check if the given string operand size is different from 0. If the length is different from 0 then it returns true. [-z $ a] is incorrect. str Check if str is not an empty string. If it is an empty string, it returns false. [$ a] is not wrong.

File checking operators in Unix / Linux

The following operators to check for basic ownership associated with a Unix **file** .

Suppose a file variable holds an existing file name as "test" with a size of 100 bytes and is allowed to read, write and run.

For example

Operator Description -b file example Check if the file is a special block file, if true, the condition is true. [-b $ file] is wrong. -c file Check if the file is a special character file, if true, the condition is true. [-c $ file] is wrong. -d file Check if the file is a directory, if true, the condition is true. [-d $ file] is incorrect. -f file Check if the file is a normal file as opposed to a directory or a special file, if true, the condition is true. [-f $ file] is correct. -g file Check if the file has a set group ID (SGID), if true, the condition is true. [-g $ file] is wrong. -k file Check if the file has sticky bit set, if it is correct, the condition is correct. [-k $ file] is wrong. -p file Check if the file is a named pipe, if true, the condition is true. [-p $ file] is wrong. -t file Check if the symbol to identify the file is opened and linked to a terminal, if true, the condition is true. [-t $ file] is wrong. -u file Check if the file has SUID, if true, the condition is correct. [-u $ file] is wrong. -r file Check if the file is readable, if true, the condition is correct. [-r $ file] is correct. -w file Check if the file is writeable, if true, the condition is correct. [-w $ file] is correct. -x file Check if the file is executable, if true, the condition is true. [-x $ file] is correct. -s file Check if the file is executable, if true, the condition is correct. [-s $ file] is correct. -e file Check if the file exists, still true even if the file is a directory but exists. [-e $ file] is correct.

C Shell operators in Unix / Linux:

The following is a brief description of the C Shell operators.

C Shell operators

Korn Shell operators in Unix / Linux:

The following is a brief description of the Korn Shell operators.

Korn Shell operators

According to Tutorialspoint

Previous article: Use array in Shell

Next lesson: C Shell operators