These are equal to `let no=no+6` and `let no=no-6` respectively.

- Alternate methods:

  The `[]` operator can be used in the same way as the `let` command as follows:

  ```
  result=$[ no1 + no2 ]
  ```

  Using the `$` prefix inside `[]` operators are legal, for example:

  ```
  result=$[ $no1 + 5 ]
  ```

  `(( ))` can also be used. `$` prefixed with a variable name is used when `(( ))` operator is used, as follows:

  ```
  result=$(( no1 + 50 ))
  ```

  `expr` can also be used for basic operations:

  ```
  result=`expr 3 + 4`
  result=$(expr $no1 + 5)
  ```

  All of the preceding methods do not support floating point numbers, and operate on integers only.

3. `bc`, the precision calculator is an advanced utility for mathematical operations. It has a wide range of options. We can perform floating point operations and use advanced functions as follows:

   ```
   echo "4 * 0.56" | bc
   2.24

   no=54;
   result=`echo "$no * 1.5" | bc`
   echo $result
   81.0
   ```

   Additional parameters can be passed to `bc` with prefixes to the operation with semicolon as delimiters through `stdin`.

   - **Decimal places scale with bc**: In the following example the `scale=2` parameter sets the number of decimal places to `2`. Hence, the output of `bc` will contain a number with two decimal places:

     ```
     echo "scale=2;3/8" | bc
     0.37
     ```

- **Base conversion with bc**: We can convert from one base number system to another one. Let us convert from decimal to binary, and binary to octal:

  ```
  #!/bin/bash
  Desc: Number conversion

  no=100
  echo "obase=2;$no" | bc
  1100100
  no=1100100
  echo "obase=10;ibase=2;$no" | bc
  100
  ```

- Calculating squares and square roots can be done as follows:

  ```
  echo "sqrt(100)" | bc #Square root
  echo "10^10" | bc #Square
  ```

Playing with file descriptors and redirection

File descriptors are integers that are associated with file input and output. They keep track of opened files. The best-known file descriptors are `stdin`, `stdout`, and `stderr`. We even can redirect the contents of one file descriptor to another. This recipe shows examples on how to manipulate and redirect with file descriptors.

Getting ready

While writing scripts we use standard input (`stdin`), standard output (`stdout`), and standard error (`stderr`) frequently. Redirection of an output to a file by filtering the contents is one of the essential things we need to perform. While a command outputs some text, it can be either an error or an output (nonerror) message. We cannot distinguish whether it is output text or an error text by just looking at it. However, we can handle them with file descriptors. We can extract text that is attached to a specific descriptor.

File descriptors are integers associated with an opened file or data stream. File descriptors 0, 1, and 2 are reserved as follows:

- 0: `stdin` (standard input)
- 1: `stdout` (standard output)
- 2: `stderr` (standard error)