The `echo` command writes a single line into the standard output. Hence, the statements in the `{ }` block of `awk` are executed once. If the standard input to `awk` contains multiple lines, the commands in `awk` will be executed multiple times.

Concatenation can be used as follows:

```
$ echo | awk '{ var1="v1"; var2="v2"; var3="v3"; \
print var1 "-" var2 "-" var3 ; }'
```

The output will be as follows:

```
v1-v2-v3
```

`{ }` is like a block in a loop, iterating through each line of a file.

Usually, we place initial variable assignments, such as `var=0;` and `like` statements, print the file header in the `BEGIN` block. In the `END{}` block, we place statements such as printing results.

There's more...

The `awk` command comes with a lot of rich features. In order to master the art of `awk` programming, you should be familiar with the important `awk` options and functionalities. Let's go through the essential functionalities of `awk`.

Special variables

Some special variables that can be used with `awk` are as follows:

- `NR`: It stands for the current record number, which corresponds to the current line number when it uses lines as records
- `NF`: It stands for the number of fields, and corresponds to the number of fields in the current record under execution (fields are delimited by space)
- `$0`: It is a variable that contains the text content of the current line under execution
- `$1`: It is a variable that holds the text of the first field
- `$2`: It is the variable that holds the text of the second field

For example:

```
$ echo -e "line1 f2 f3\nline2 f4 f5\nline3 f6 f7" | \

awk '{
```

```
print "Line no:"NR",No of fields:"NF, "$0="$0, "$1="$1,"$2="$2,"$3="$3
}'
Line no:1,No of fields:3 $0=line1 f2 f3 $1=line1 $2=f2 $3=f3
Line no:2,No of fields:3 $0=line2 f4 f5 $1=line2 $2=f4 $3=f5
Line no:3,No of fields:3 $0=line3 f6 f7 $1=line3 $2=f6 $3=f7
```

We can print the last field of a line as `print $NF`, last but the second as `$(NF-1)`, and so on.

`awk` also provides the `printf()` function with the same syntax as in C. We can also use that instead of print.

Let's see some basic `awk` usage examples. Print the second and third field of every line as follows:

```
$awk '{ print $3,$2 }'  file
```

In order to count the number of lines in a file, use the following command:

```
$ awk 'END{ print NR }' file
```

Here, we only use the `END` block. `NR` will be updated on entering each line by `awk` with its line number. When it reaches the end of the line, it will have the value of the last line number. Hence, in the `END` block `NR` will have the value of the last line number.

You can sum up all the numbers from each line of `field 1` as follows:

```
$ seq 5 | awk 'BEGIN{ sum=0; print "Summation:" }
{ print $1"+"; sum+=$1 } END { print "=="; print sum }'
Summation:
1+
2+
3+
4+
5+
==
15
```

Passing an external variable to awk

By using the `-v` argument, we can pass external values other than `stdin` to `awk`, as follows:

```
$ VAR=10000
$ echo | awk -v VARIABLE=$VAR '{ print VARIABLE }'
10000
```