Using the `%` operator, we have:

```
$ echo ${VAR%.*}
```

The output will be: `hack.fun.book`.

The `%` operator performs a nongreedy match for `.*` from right to left (`.txt`).

Using the `%%` operator, we have:

```
$ echo ${VAR%%.*}
```

The output will be: `hack`

The `%%` operator performs a greedy match for `.*` going right to left (`.fun.book.txt`).

In the second task, we have used the # operator to extract the extension from the filename. It is similar to `%`. But it evaluates from left to right.

`${VAR#*.}` can be interpreted as:

- Remove the string match from `$VARIABLE` for the wildcard pattern match appears to the right-hand side of the # (`*.` in the previous example). Evaluating from the left to right direction should make the wildcard match.

Similarly, as in the case of `%%`, we have another greedy operator for #, which is ##.

It makes greedy matches by evaluating from left to right and removes the match string from the specified variable.

Let's use this example:

```
VAR=hack.fun.book.txt
```

By using the # operator, we have:

```
$ echo ${VAR#*.}
```

The output will be: `fun.book.txt`.

The # operator performs a nongreedy match for `*.` from left to right (`hack.`).

By using the ## operator, we have:

```
$ echo ${VAR##*.}
```

The output will be: `txt`.

The ## operator matches a greedy match for *. from left to right (`txt`).

The ## operator is preferred over the # operator to extract the extension from a filename since the filename may contain multiple "." characters. Since ## makes a greedy match, it always extracts extensions only.

Here is practical example that can be used to extract different portions of a domain name, given a `URL="www.google.com"`:

```
$ echo ${URL%.*} # Remove rightmost .*
www.google

$ echo ${URL%%.*} # Remove right to leftmost  .* (Greedy operator)
www


$ echo ${URL#*.} # Remove leftmost  part before *.
google.com


$ echo ${URL##*.} # Remove left to rightmost  part before *. (Greedy
operator)
com
```

Renaming and moving files in bulk

Renaming a number of files is one of the tasks we frequently come across. A simple example is when you download photos from your digital camera to your computer you may delete unnecessary files and it causes discontinuous numbering of image files. Sometimes, you may need to rename them with a custom prefix and continuous numbering for filenames. We sometimes use third-party tools for performing rename operations. We can use Bash commands to perform a rename operation in a couple of seconds.

Moving all the files having a particular substring in their filenames (for example, the same prefix for filenames) or with a specific file type to a given directory is another use case we frequently perform. Let's see how to write scripts to perform these kinds of operations.