Bash has a powerful parameter syntax. The examples below show a command on the left, the result (if any) in the center, and an explanation on the right. See “Parameter Expansion” in the bash man page.

Set Up the Test Directory

```
$ mkdir test
$ cd test
$ touch file1 file2 file3
```

Basic Parameter Expansion

```
$ param1=hello
$ echo $param1             hello
$ echo ${param1}a          helloa                  the braces separate the name
$ echo ${param2}                                   nothing there
```

Default Values

```
$ echo ${param2:-file*}     file1 file2 file3       all files in the directory
$ echo ${param2:-$param1}   hello                   uses $param1's value...
$ echo $param2              (nothing)               ... but didn't change $param2
$ echo ${param3:=$param1}   hello                   uses $param1's value...
$ echo $param3              hello                   ... and assigns it to $param3
```

Substring Manipulations

Keep in mind that the substitutions etc are expanded, not literals, so you can use wildcards and other pattern syntaxes in them (for example, the “he*l” below used to strip “hell” from the value).

```
$ echo ${param1:2}          llo                     substring from 2
$ echo ${param1:2:2}        ll                      substring from 2, len 2
$ echo ${param1#he}         llo                     strip shortest match from start
$ echo ${param1#hel*}       lo                      strip shortest match from start
$ echo ${param1#he*l}       lo                      strip shortest match from start
$ echo ${param1##he*l}      o                       strip longest match from start
$ echo ${param1%l*o}        hel                     strip shortest match from end
$ echo ${param1%%l*o}       he                      strip longest match from end
$ echo ${param1/l/p}        heplo                   replace as few as possible
$ echo ${param1//l/p}       heppo                   replace as many as possible
```

Miscellaneous

```
$ echo ${!param*}           param1 param2 param3    parameter names starting with...
$ echo ${#param1}           5                       length of parameter value
```

Example Uses

```
# Rename all .GIF files to .gif
for file in *.GIF; do mv $file ${file%GIF}gif; done

# Now number the files sequentially
cnt=0;
for file in *.gif; do mv $file $cnt$file; let cnt=cnt++; done

# Oops, I didn't mean that... get rid of the numbers.
for file in *.gif; do mv $file ${file##[0-9]}; done
```