```
[connection] 192.168.0.2 pending
[disconnect] 192.168.0.3 pending
[connection] 192.168.0.4 failed
```

We may need to split the files into `server1.log`, `server2.log`, and `server3.log` from the contents for each `SERVER` in each file. This can be done as follows:

```
 $ csplit server.log /SERVER/ -n 2 -s {*}  -f server -b "%02d.log"  ; rm
server00.log


$ ls
server01.log  server02.log  server03.log  server.log
```

The details of the command are as follows:

- `/SERVER/` is the line used to match a line by which a split is to be carried out.
- `/[REGEX]/` is the format. It copies from the current line (first line) up to the matching line that contains `"SERVER"` excluding the match line.
- `{*}` is used to specify to repeat a split based on the match up to the end of the file. By using `{integer}`, we can specify the number of times it is to be continued.
- `-s` is the flag to make the command silent rather than printing other messages.
- `-n` is used to specify the number of digits to be used as suffix. 01, 02, 03, and so on.
- `-f` is used for specifying the filename prefix for split files (`server` is the prefix in the previous example).
- `-b` is used to specify the suffix format. `"%02d.log"` is similar to the `printf` argument format in C. Here, the filename = prefix + suffix, that is, `"server" + "%02d.log"`.

We remove `server00.log` since the first split file is an empty file (the match word is the first line of the file).

Slicing filenames based on extension

Several shell scripts perform manipulations based on filenames. We may need to perform actions such as renaming the files by preserving the extension, converting files from one format to another (change the extension by preserving the name), extracting a portion of the filename, and so on. The shell comes with inbuilt features for slicing filenames based on different conditions. Let us see how to do it.

How to do it...

The name from `name.extension` can be easily extracted using the `%` operator. You can extract the name from `"sample.jpg"` as follows:

```
file_jpg="sample.jpg"
name=${file_jpg%.*}
echo File name is: $name
```

The output is:

```
File name is: sample
```

The next task is to extract the extension of a file from its filename. The extension can be extracted using the # operator as follows:

Extract `.jpg` from the filename stored in the variable `file_jpg` as follows:

```
extension=${file_jpg#*.}
echo Extension is: jpg
```

The output is:

```
Extension is: jpg
```

How it works...

In the first task, in order to extract the name from the filename in the format `name.extension` we have used the `%` operator.

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

- Remove the string match from the `$VAR` for the wildcard pattern that appears to the right-hand side of `%` (`.*` in the previous example). Evaluating from the right to left direction should make the wildcard match.
- Let's store the filename as `VAR=sample.jpg`. Therefore, the wildcard match for `.*` from right to left is `.jpg`. Thus, it is removed from the `$VAR` string and the output will be `sample`.

`%` is a nongreedy operation. It finds the minimal match for the wildcard from right to left. There is an operator `%%`, which is similar to `%`. But it is greedy in nature. This means, it finds the maximal match of the string for the wildcard. For example, we have:

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