- Removes extra spaces:

  ```
  tr -s ' ' or sed 's/[ ]\+/ /g'
  ```

- Removes comments:

  ```
  sed 's:/\*.*\*/::g'
  ```

  `:` is used as a `sed` delimiter to avoid the need of escaping `/` since we need to use `/*` and `*/`.

  in sed, `*` is escaped as `\*`.

  `.*` is used to match all the text in between `/*` and `*/`.

- Removes all the spaces preceding and suffixing the {, }, `(`, `)`, `;`, `:`, and `,`.

  ```
  sed 's/ \?\([{}();,:]\) \?/\1/g'
  ```

  The preceding `sed` statement works as follows:

  - `/ \?\([{}();,:]\) \?/` in the `sed` code is the match part, and `/\1 /g` is the replacement part.
  - `\([{}();,:]\)` is used to match any one character in the set `[ { }( ) ; , : ]` (inserted spaces for readability). `\(` and `\)` are group operators used to memorize the match and back reference in the replacement part. `(` and `)` are escaped to give them a special meaning as a group operator.`\?` precedes and follows the group operators. It is to match the space character that may precede or follow any of the characters in the set.
  - In the replacement part, the match string (that is, the combination of `:`, a space (optional), a character from the set, and again an optional space) is replaced with the character matched. It uses a back reference to the character matched and memorized using the group operator `()`. Back-referenced characters refer to a group match using the `\1` symbol.

The decompression command works as follows:

- `s/;/;\n/g` replaces `;` with `;\n`
- `s/{/{\n\n/g` replaces { with {\n\n
- `s/}/\n\n}/g` replaces } with \n\n}

## See also

- The *Using sed to perform text replacement* recipe in this chapter explains the `sed` command
- The *Translate with tr* recipe in *Chapter 2*, *Have a Good Command*, explains the `tr` command

# Merging multiple files as columns

There are different cases when we need to concatenate files by their columns. We may need each file's content to appear in separate columns. Usually, the `cat` command concatenates in a line (or row-wise) fashion.

## How to do it...

`paste` is the command that can be used for column-wise concatenation. The `paste` command can be used with the following syntax:

```
$ paste file1 file2 file3 …
```

Let's try an example as follows:

```
$ cat file1.txt
1
2
3
4
5
$ cat file2.txt
slynux
gnu
bash
hack
$ paste file1.txt file2.txt
1slynux
2gnu
3bash
4hack
5
```

The default delimiter is tab. We can also explicitly specify the delimiter by using `-d`. For example:

```
$ paste file1.txt file2.txt -d ","
1,slynux
2,gnu
3,bash
4,hack
5,
```