Using sed to perform text replacement

`sed` stands for **stream editor**. It is a very essential tool for text processing, and a marvelous utility to play around with regular expressions. A well-known usage of the `sed` command is for text replacement. This recipe will cover most of the frequently-used `sed` techniques.

How to do it...

`sed` can be used to replace occurrences of a string with another string in a given text.

1. It can be matched using regular expressions.

   ```
   $ sed 's/pattern/replace_string/' file
   ```

   Or:

   ```
   $ cat file | sed 's/pattern/replace_string/'
   ```

   This command reads from `stdin`.

   

   If you use the `vi` editor, you will notice that the command to replace the text is very similar to the one discussed here.

2. By default, `sed` only prints the substituted text. To save the changes along with the substitutions to the same file, use the `-i` option. Most of the users follow multiple redirections to save the file after making a replacement as follows:

   ```
   $ sed 's/text/replace/' file >newfile
   $ mv newfile file
   ```

   However, it can be done in just one line; for example:

   ```
   $ sed -i 's/text/replace/' file
   ```

3. These usages of the `sed` command will replace the first occurrence of the pattern in each line. If we want to replace every occurrence, we need to add the `g` parameter at the end, as follows:

   ```
   $ sed 's/pattern/replace_string/g' file
   ```

   The `/g` suffix means that it will substitute every occurrence. However, we sometimes need to replace only the Nth occurrence onwards. For this, we can use the `/Ng` form of the option.

Have a look at the following commands:

```
$ echo thisthisthisthis | sed 's/this/THIS/2g'
thisTHISTHISTHIS

$ echo thisthisthisthis | sed 's/this/THIS/3g'
thisthisTHISTHIS

$ echo thisthisthisthis | sed 's/this/THIS/4g'
thisthisthisTHIS
```

We have used `/` in `sed` as a delimiter character. We can use any delimiter characters as follows:

```
sed 's:text:replace:g'
sed 's|text|replace|g'
```

When the delimiter character appears inside the pattern, we have to escape it using the `\` prefix, as follows:

```
sed 's|te\|xt|replace|g'
```

`\|` is a delimiter appearing in the pattern replaced with escape.

There's more...

The `sed` command comes with numerous options for text manipulation. By combining the options available with `sed` in logical sequences, many complex problems can be solved in one line. Let's see the different options available with `sed`.

Removing blank lines

Removing blank lines is a simple technique by using `sed` to remove blank lines. Blanks can be matched with regular expression `^$`:

```
$ sed '/^$/d' file
```

`/pattern/d` will remove lines matching the pattern.

For blank lines, the line end marker appears next to the line start marker.

Performing replacement directly in the file

When a filename is passed to `sed`, it usually prints its output to `stdout`. Instead, if we want it to actually modify the contents of the file, we use the `-i` option, as follows:

```
$ sed 's/PATTERN/replacement/' -i filename
```