STREAM EDITOR - MANAGING PATTERNS

We have already discussed the use of pattern and hold buffer. In this chapter, we are going to explore more about their usage. Let us discuss the **n** command which prints the pattern space. It will be used in conjunction with other commands. Given below is the syntax of the**n** command.

```
[address1[,address2]]n
```

Let us take an example.

```
[jerry]$ sed 'n' books.txt
```

When the above code is executed, it will produce the following result:

```
1) A Storm of Swords, George R. R. Martin, 1216
2) The Two Towers, J. R. R. Tolkien, 352
3) The Alchemist, Paulo Coelho, 197
4) The Fellowship of the Ring, J. R. R. Tolkien, 432
5) The Pilgrimage, Paulo Coelho, 288
6) A Game of Thrones, George R. R. Martin, 864
```

The **n** command prints the contents of the pattern buffer, clears the pattern buffer, fetches the next line into the pattern buffer, and applies commands on it.

Let us consider there are three SED commands before **n** and two SED commands after **n** as follows:

```
Sed command #1
Sed command #2
Sed command #3
n command
Sed command #4
Sed command #5
```

In this case, SED applies the first three commands on the pattern buffer, clears the pattern buffer, fetches the next line into the pattern buffer, and thereafter applies the fourth and fifth commands on it. This is a very important concept. Do not go ahead without having a clear understanding of this.

The hold buffer holds data, but SED commands cannot be applied directly on the hold buffer. Hence, we need to bring the hold buffer data into the pattern buffer. SED provides the **x** command to exchange the contents of pattern and hold buffers. The following commands illustrate the **x** command.

Let us slightly modify the books.txt file. Say, the file contains book titles followed by their author names. After modification, the file should look like this:

```
[jerry]$ cat books.txt
```

On executing the above code, you get the following result:

```
A Storm of Swords
George R. R. Martin
The Two Towers
J. R. R. Tolkien
The Alchemist
Paulo Coelho
The Fellowship of the Ring
J. R. R. Tolkien
The Pilgrimage
Paulo Coelho
```

```
A Game of Thrones
George R. R. Martin
```

Let us exchange the contents of the two buffers. For instance, the following example prints only the names of authors.

```
[jerry]$ sed -n 'x;n;p' books.txt
```

On executing the above code, you get the following result:

```
George R. R. Martin
J. R. R. Tolkien
Paulo Coelho
J. R. R. Tolkien
Paulo Coelho
George R. R. Martin
```

Let us understand how this command works.

- Initially, SED reads the first line, i.e., A Storm of Swords into the pattern buffer.
- **x** command moves this line to the hold buffer.
- **n** fetches the next line, i.e., George R. R. Martin into the pattern buffer.
- The control passes to the command followed by n which prints the contents of the pattern buffer.
- The process repeats until the file is exhausted.

Now let us exchange the contents of the buffers before printing. Guess, what happens? Yes, it prints the titles of books.

```
[jerry]$ sed -n 'x;n;x;p' books.txt
```

On executing the above code, you get the following result:

```
A Storm of Swords
The Two Towers
The Alchemist
The Fellowship of the Ring
The Pilgrimage
A Game of Thrones
```

The **h** command deals with the hold buffer. It copies data from the pattern buffer to the hold buffer. Existing data from the hold buffer gets overwritten. Note that the **h** command does not move data, it only copies data. Hence, the copied data remains as it is in the pattern buffer. Given below is the syntax of the **h** command.

```
[address1[,address2]]h
```

The following command prints only the titles of the author Paulo Coelho.

```
[jerry]$ sed -n '/Paulo/!h; /Paulo/{x;p}' books.txt
```

On executing the above code, you get the following result:

```
The Alchemist
The Pilgrimage
```

Let us understand how the above command works. The contents of books.txt follow a specific format. The first line is the book title followed by the author of the book. In the above command, "!" is used to reverse the condition, i.e., line is copied to the hold buffer only when a pattern match

does not succeed. And curly braces {} are used to group multiple SED commands

In the first pass of the command, SED reads the first line, i.e., A Storm of Swords into the pattern buffer and checks whether it contains the pattern Paulo or not. As the pattern match does not succeed, it copies this line to the hold buffer. Now both the pattern buffer and the hold buffer contain the same line i.e., A Storm of Swords. In the second step, it checks whether the line contains the pattern Paulo or not. As the pattern does not match, it does not do anything.

In second pass, it reads the next line George R. R. Martin into the pattern buffer and applies the same steps. For the next three lines, it does the same thing. At the end of the fifth pass, both the buffers contain The Alchemist. At the start of the sixth pass, it reads the line Paulo Coelho and as the pattern matches, it does not copy this line into the hold buffer. Hence, the pattern buffer contains Paulo Coelho, and the hold buffer contains The Alchemist.

Thereafter, it checks whether the pattern buffer contains the pattern Paulo. As the pattern match succeeds, it exchanges the contents of the pattern buffer with the hold buffer. Now the pattern buffer contains The Alchemist and the hold buffer contains Paulo Coelho. Finally, it prints the contents of the pattern buffer. The same steps are applied to the pattern The Pilgrimage.

The **h** command destroys the previous contents of the hold buffer. This is not always acceptable, as sometimes we need to preserve the contents. For this purpose, SED provides the **H** command which appends the contents to the hold buffer by adding a new line at the end. The only difference between **h** and **H** command is, the former overwrites data from the hold buffer, while the later appends data to the hold buffer. Its syntax is similar to the **h** command.

```
[address1[,address2]]H
```

Let us take another example. This time, instead of printing only book titles, print the names of their authors too. The following example prints the book titles followed by their author names.

```
[jerry]$ sed -n '/Paulo/!h; /Paulo/{H;x;p}' books.txt
```

On executing the above code, you get the following result:

```
The Alchemist
Paulo Coelho
The Pilgrimage
Paulo Coelho
```

We learnt how to copy/append the contents of pattern buffer to hold buffer. Can we perform the reverse function as well? Yes certainly! For this purpose, SED provides the **g** command which copies data from the hold buffer to the pattern buffer. While copying, existing data from the pattern space gets overwritten. Given below is the syntax of the **g** command.

```
[address1[,address2]]g
```

Let us consider the same example - printing book titles and their authors. This time, we will first print the name of the author and on the next line, the corresponding book title. The following command prints the name of the author Paulo Coelho, followed by its book title.

```
[jerry]$ sed -n '/Paulo/!h; /Paulo/{p;g;p}' books.txt
```

On executing the above code, you get the following result:

```
Paulo Coelho
The Alchemist
Paulo Coelho
The Pilgrimage
```

The first command is kept as it is. At the end of fifth pass, both the buffers contain The Alchemist. At the start of the sixth pass, it reads the line Paulo Coelho and as the pattern matches, it does not copy this line into the hold buffer. Hence, the pattern space containsPaulo Coelho and the hold space contains The Alchemist.

Thereafter, it checks whether the pattern space contains the pattern Paulo. As the pattern match succeeds, it first prints the contents of the pattern space, i.e., Paulo Coelho, then it copies the hold buffer to the pattern buffer. Hence, both the pattern and hold buffers contain The Alchemist. Finally, it prints the contents of the pattern buffer. The same steps are applied to the pattern The Pilgrimage.

Similarly, we can append the contents of the hold buffer to the pattern buffer. SED provides the **G** command which appends the contents to the pattern buffer by adding a new line at the end.

```
[address1[,address2]]G
```

Now let us take the previous example which prints the name of author Paulo Coelhofollowed by its book title. To achieve the same result, execute the following SED command.

```
[jerry]$ sed -n '/Paulo/!h; /Paulo/{G;p}' books.txt
```

On executing the above code, you get the following result:

```
Paulo Coelho
The Alchemist
Paulo Coelho
The Pilgrimage
```

Can you modify the above example to display the book titles followed by their authors? Simple, just exchange the buffer contents before the **G** command.

```
[jerry]$ sed -n '/Paulo/!h; /Paulo/{x;G;p}' books.txt
```

On executing the above code, you get the following result:

```
The Alchemist
Paulo Coelho
The Pilgrimage
Paulo Coelho
```