RegularExpressionsWithSED

April 15, 2024

1 What is regular expression(re) in Shell Scripting?

A regular expression (regex) is a sequence of characters that forms a search pattern. It's used for pattern matching with strings, or string matching, i.e. "find this pattern within this string".

In shell scripting, regular expressions are used in several commands to search, replace, and compare text. Some of these commands include `grep`, `sed`, `awk`, and `perl`.

Here are some basic regular expression symbols:

- `.`: Matches any single character.
- `*`: Matches zero or more of the preceding character or pattern.
- `^`: Matches the beginning of the line.
- `$`: Matches the end of the line.
- `[abc]`: Matches any one of the enclosed characters.
- `[^abc]`: Matches any character not enclosed.
- `[a-z]`: Matches any character in the range a-z.
- `\`: Escapes special characters.

Here's an example of using a regular expression with the `grep` command:

[ ]:
```
grep '^abc' file.txt
```

This command will print all lines in `file.txt` that start with 'abc'.

Regular expressions are a powerful tool for manipulating text in shell scripting. They have a complex syntax that allows for very flexible and precise pattern matching.

2 What is SED?

`sed`, short for Stream Editor, is a powerful utility in Unix-like operating systems. It's used for parsing and transforming text.

`sed` operates on a line-by-line basis, applying instructions specified in a script to each line in the input. The input can be piped from another command or read from a file.

Here are some common uses of `sed`:

1. **Substitution**: The most common use of `sed` is to replace one string with another.

   ```
   sed 's/foo/bar/' file.txt  # Replace the first occurrence of 'foo' with 'bar' in each line
   ```

2. **Deletion**: `sed` can delete lines that match a certain pattern.

```
sed '/foo/d' file.txt  # Delete all lines containing 'foo'
```

3. **Insertion and Appending**: `sed` can insert or append lines of text.

```
sed '1i\This is the start' file.txt  # Insert a line at the beginning of the file
sed '$a\This is the end' file.txt  # Append a line at the end of the file
```

4. **File Splicing and Joining**: `sed` can read and append multiple files together.

```
sed 'R another_file.txt' file.txt  # For each line in file.txt, append a line from another
```

`sed` commands can be combined into a script file, which can be run with `sed -f script.sed file.txt`.

While `sed` has a somewhat complex syntax, it's extremely powerful for text processing and manipulation in shell scripts.

3 Deleting All Lines with sed?

You can use `sed` to delete all lines in a file. Here's how you can do it:

[ ]:
```
sed -i 'd' file.txt
```

In this command:

- `-i` option tells `sed` to edit files in place (i.e. save back to the original file).
- `'d'` is the `sed` command to delete all lines.
- `file.txt` is the file you want to delete all lines from.

Please note that this command will delete all lines from the file. If you want to delete lines matching a specific pattern, you can use the following command:

[ ]:
```
sed -i '/pattern_to_match/d' file.txt
```

This command will delete all lines containing `pattern_to_match` from `file.txt`.

4 The sed Addresses?

In `sed`, an address is a way to specify which lines of the input you want a command to apply to. Addresses can be specified in several ways:

1. **By line number**: You can specify a single line number, or a range of line numbers.

```
sed '3d' file.txt  # Delete the third line
sed '2,4d' file.txt  # Delete the second through fourth lines
```

2. **By pattern**: You can specify a regular expression pattern, and the command will apply to lines that match the pattern.

```
sed '/foo/d' file.txt  # Delete lines containing 'foo'
```

3. **By last line**: The `$` character represents the last line of the input.

```
sed '$d' file.txt  # Delete the last line
```

4. **By interval**: You can specify an interval with `~`.

   ```
   sed '1~2d' file.txt  # Delete every other line starting from the first line
   ```

5. **Multiple addresses**: You can specify multiple separate addresses or ranges.

   ```
   sed -e '3d' -e '5,7d' file.txt  # Delete the third line and lines five through seven
   ```

In all these examples, `d` is the `sed` command to delete a line. You can replace it with any other `sed` command.

5 Thank You!