`head` and `tail` can also be used in pipelines:

```
$ cut -d, -f1 < parking_data.csv | sort | uniq | head --n 5
```

```
$ cut -d, -f1 < parking_data.csv | sort | unig | tail -n 5
```

Expansions

Expansion uses special characters to expand upon something before the shell processes it. We have learned a few expansions so far such as the tilde `~` and wildcards `*`. We've also seen some character wildcards `[characters]`.

Expansions are another feature that help us when we're manipulating and working with files and directories.

Other examples of expansions are:

- arithmetic expansion
- brace expansion

1. `${parameter#pattern}` removes the shortest leading portion of the string contained in *parameter* defined by the *pattern*.

```
$ foo=/User/name/Desktop/file.txt.zip
$ echo ${foo#/x/}
```

In this example, we've defined foo as a file with an extension.

The expansion matches any (`*`) pattern of `/*/` and returns the shortest leading portion.

2. `${parameter##pattern}` is very similar to the previous expansion except it removes the longest leading portion of the string.

```
$ foo=/User/name/Desktop/file.txt.zip
$ echo ${foo##/x/}
```

Very similar to the previous example, the expansion matches any (`*`) pattern of `/*/` and returns the longest leading portion.

3. `${parameter%spattern}` removes the shortest ending portion of the string rather than the beginning.

```
$ foo=/User/name/Desktop/file.txt.zip
$ echo ${fo00%.x}
```

```
while condition; do
    commands
done
```

Let's make a basic while script that displays five numbers in sequential order from 1 to 5 and then tells us when it's finished.

```
#!/bin/bash

# script called while-count.sh

count=1

while [ $count -le 5 ]; do
    echo $count
    count=$((count +1))
done
echo "Finished."
```

Why does the loop end?

While loops are extremely helpful to read lines of a file and then perform some command if