{:title “Clojure - Shell Commands” :layout :post :date “2016-10-23” :tags [“Clojure” “Code” “Guide”]}

Sometimes it is nice to be able to do shell commands right from the program. This can be done with clojure.java.shell. So go ahead and require or use that in the file your going to be using.

```
(use 'clojure.java.shell)
```

Now that you have that go ahead and try the following...

```
(sh "ls")
```

You'll notice that you should get a persistant Array Map. The contents of this array map include :exit, :out, and :err. You will probably be most interested with the :out since that is the standard output of the command issued. However you do also have :exit and :err which can come in handly if anything happens that wasn't expected. Notice that if you try to do two commands like the following.

```
(sh "ls -l")
```

You will get an error this is because sh takes multiple arguments and so you have seperate the spaces and do something like this.

```
(sh "ls" "-l")
```

So now we get the result we want but I dont want to get the :exit or :err so we can build a small framework to make it easier for us to just see the standard output. So we can do this for just the output.

```
(:out (sh "ls" "-l"))
```

However that doesnt' look that great so lets get a few more things such as pprint and split from clojure.pprint and clojure.string.

```
(use 'clojure.pprint)
(use 'clojure.string)
```

This will allow us now to split the string that is handed to us from the :out of the sh function based on regex. As well as the pprint which will pprint out it out nicely.

```
(pprint (split (:out (sh "ls" "-l")) #"\n"))
```

Now you have a much nicer string to look at and looks a lot like you did ls -l on a terminal if you ignore the two brackets.