# grep Command

coursera.org/learn/linux-tools-for-developers/supplement/4wdj3/grep-command

**grep** is a workhorse command line utility whose basic job is to search files for patterns and print out matches according to specified options.

Its name stands for global regular expression print, which points out that it can do more than just match simple strings; it can work with more complicated regular expressions which can contain wildcards and other special attributes.

The simplest example of using **grep** would be:

```
5

1

2

3

4

pig food

$ grep pig file


pig

dirty pig
```

which finds three instances of the string "pig" in file.

As an example:

```
1

$ grep -i -e pig -e dog -r .
```

will search all files in the current directory and those below it for the strings "pig" or "dog", ignoring case.

If we try to explore the use of regular expressions in detail, it would be a large topic, but here are some examples:

```
1
2
3
4
5
6
7
8
# print all lines that start with "dog"
$ grep "^dog" file

# print all lines that end with "dog"
$ grep "dog$" file

# print all lines that end with "dog"
$ grep d[a-p] file
```

**grep** has many options; some of the most important are:

| Option | Meaning |
|---|---|
| **-i** | Ignore case |
| **-v** | Invert match |
| **-n** | Print line number |

| Option | Meaning |
| --- | --- |
| **-H** | Print filename |
| **-a** | Treat binary files as text |
| **-I** | Ignore binary files |
| **-r** | Recurse through subdirectories |
| **-l** | Print out names of all files that contain matches |
| **-L** | Print out names of all files that do not contain matches |
| **-c** | Print out number of matching lines only |
| **-e** | Use the following pattern; useful for multiple strings and special characters |

## Completed