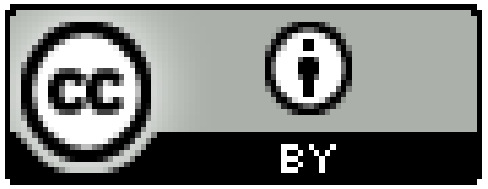
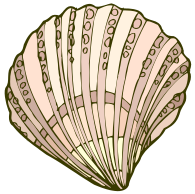
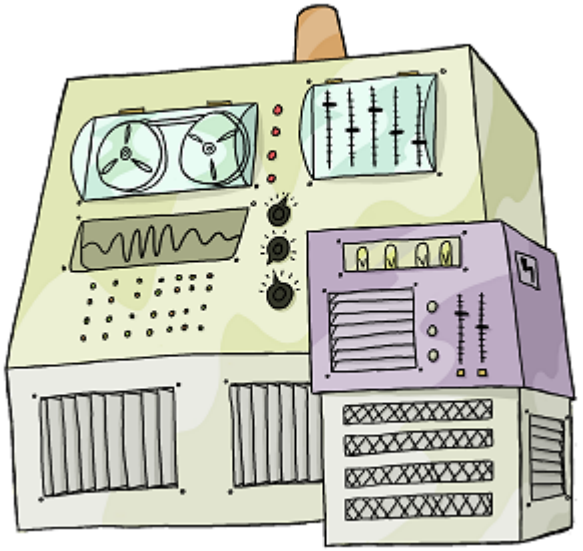
The Unix Shell

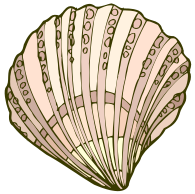
Finding Things



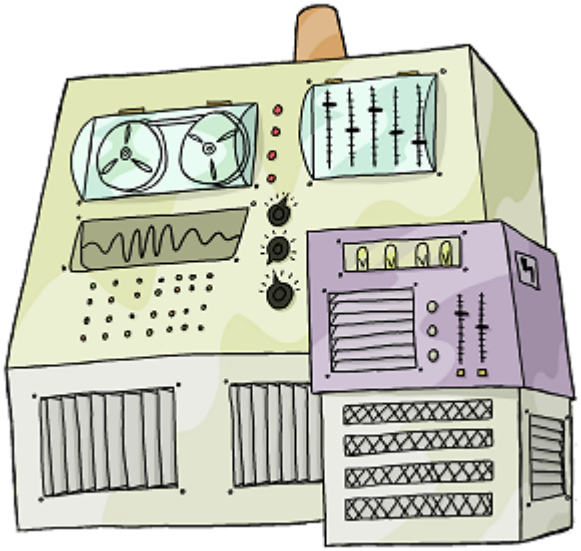


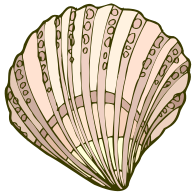
shell





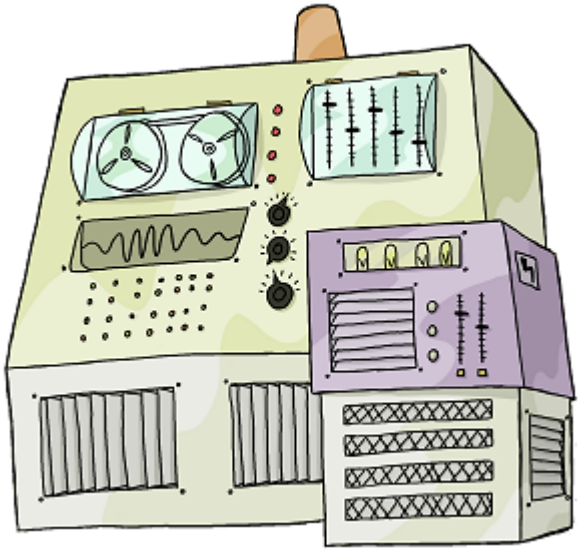
shell

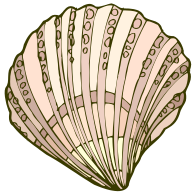




shell

Let's Google for that

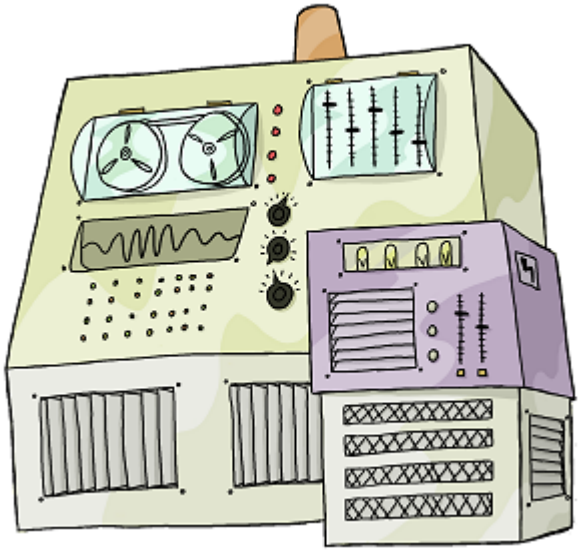




shell

Let's grep

for that



grep: global / regular expression / print

grep: global / regular expression / print

Finds and prints lines in files that match a pattern

grep: global / regular expression / print

Finds and prints lines in files that match a pattern

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence:
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

haiku.txt

grep: global / regular expression / print

Finds and prints lines in files that match a pattern

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence:
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

```
$ grep not haiku.txt
```

```
haiku.txt
```

grep: global / regular expression / print

Finds and prints lines in files that match a pattern

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence:
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

haiku.txt

```
$ grep not haiku.txt
```

Pattern

grep: global / regular expression / print

Finds and prints lines in files that match a pattern

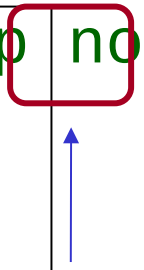
```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence:
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

haiku.txt

```
$ grep not haiku.txt
```



Pattern

Every letter matches itself

grep: global / regular expression / print

Finds and prints lines in files that match a pattern

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence:
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

```
$ grep not haiku.txt
```

File(s)

```
haiku.txt
```

grep: global / regular expression / print

Finds and prints lines in files that match a pattern

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence:
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

haiku.txt

```
$ grep not haiku.txt
Is not the true Tao, until
"My Thesis" not found
Today it is not working
$
```

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

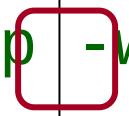
```
$ grep day haiku.txt
Yesterday it worked
Today it is not working
$
```

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

```
$ grep day haiku.txt
Yesterday it worked
Today it is not working
$ grep -w day haiku.txt
$
```



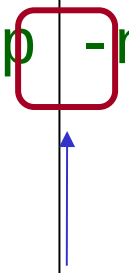
Match whole words

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

```
$ grep day haiku.txt
Yesterday it worked
Today it is not working
$ grep -w day haiku.txt
$ grep -n it haiku.txt
```



Prefix matches with line numbers

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

```
$ grep day haiku.txt
Yesterday it worked
Today it is not working
$ grep -w day haiku.txt
$ grep -n it haiku.txt
5:With searching comes loss
9:Yesterday it worked
10:Today it is not working
$
```

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

```
$ grep day haiku.txt
Yesterday it worked
Today it is not working
$ grep -w day haiku.txt
$ grep -n it haiku.txt
5:With searching comes loss
9:Yesterday it worked
10:Today it is not working
$ grep -w -n it haiku.txt
```

Use multiple flags

to combine effects

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

```
$ grep day haiku.txt
Yesterday it worked
Today it is not working
$ grep -w day haiku.txt
$ grep -n it haiku.txt
5:With searching comes loss
9:Yesterday it worked
10:Today it is not working
$ grep -w -n it haiku.txt
9:Yesterday it worked
10:Today it is not working
$
```

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence.
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

```
$ grep -i -v the haiku.txt
You bring fresh toner.

With searching comes loss

Yesterday it worked
Today it is not working
Software is like that.
$
```

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence.
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

```
$ grep -i -v the haiku.txt
You bring fresh toner.

With searching comes loss

Yesterday it worked
Today it is not working
Software is like that.
$
```

-i case insensitive

```
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence.
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

```
$ grep -i -v the haiku.txt
You bring fresh toner.

With searching comes loss

Yesterday it worked
Today it is not working
Software is like that.
$
```

-i case insensitive

-v invert and print

non-matches

Many more options

Many more options

Use `man grep` to get help

Many more options

Use `man` `grep` to get help

Many more options

Use `man grep` to get help

Complex patterns use regular expressions

Many more options

Use `man grep` to get help

Complex patterns use regular expressions

(The 're' in `grep`)

Many more options

Use `man grep` to get help

Complex patterns use regular expressions

(The 're' in `grep`)

Ideas are covered in a separate lecture

Many more options

Use `man grep` to get help

Complex patterns use regular expressions

(The 're' in `grep`)

Ideas are covered in a separate lecture

`grep`'s regular expressions are slightly different

from those provided in most programming languages

Many more options

Use `man grep` to get help

Complex patterns use regular expressions

(The 're' in `grep`)

Ideas are covered in a separate lecture

`grep`'s regular expressions are slightly different

from those provided in most programming languages

But the ideas are the same

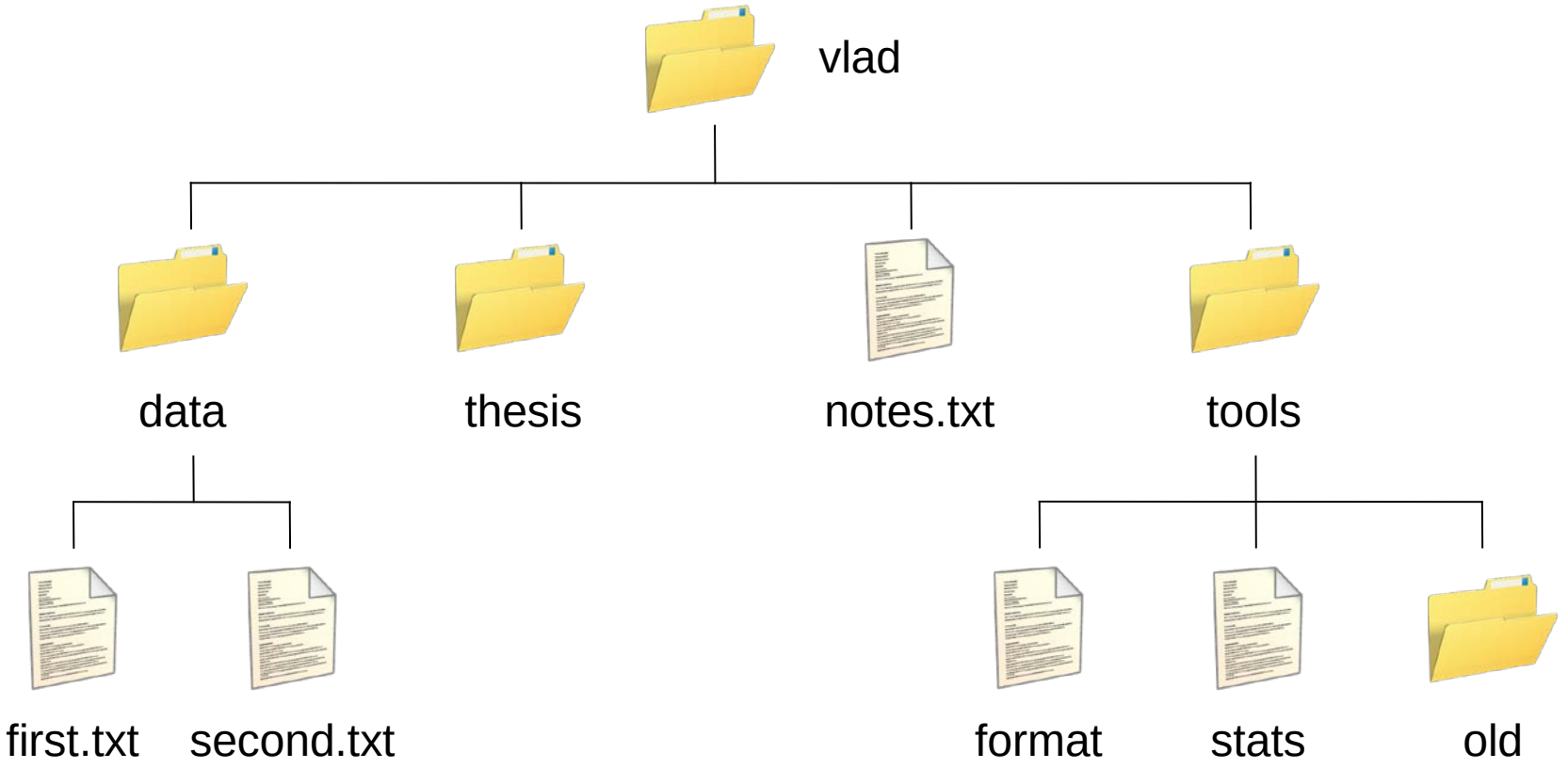
`find`: finds files (rather than lines in files)

`find`: finds files (rather than lines in files)

Again, too many options to cover here

`find`: finds files (rather than lines in files)

Again, too many options to cover here



`find`: finds files (rather than lines in files)

Again, too many options to cover here

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

`find`: finds files (rather than lines in files)

Again, too many options to cover here

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

Trailing `/` shows directories

`find`: finds files (rather than lines in files)

Again, too many options to cover here

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

Trailing `/` shows directories

Trailing `*` shows executables

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -type d
```

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -type d
```

Root directory of search

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -type d
```

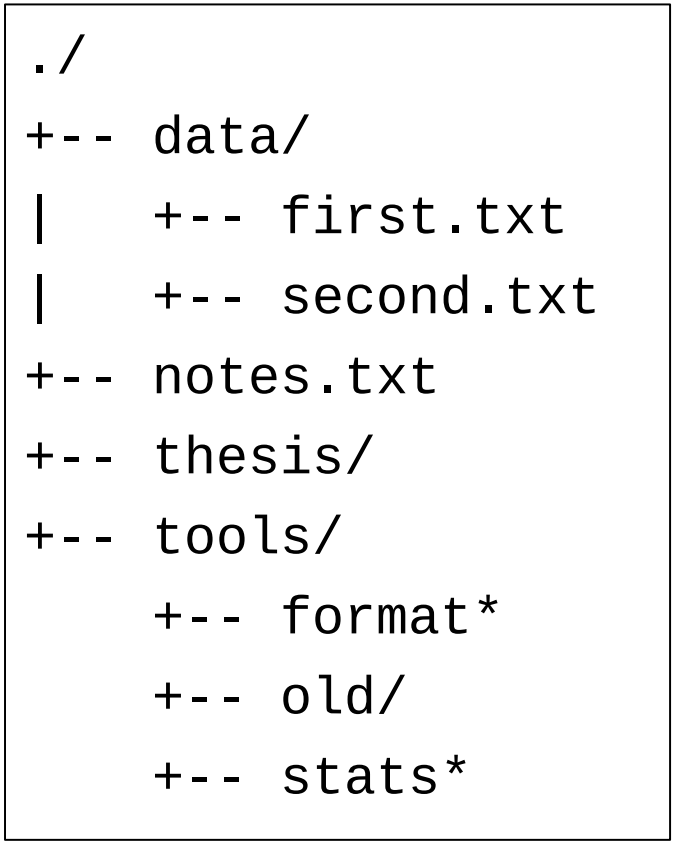
Things of type 'd' (directory)

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

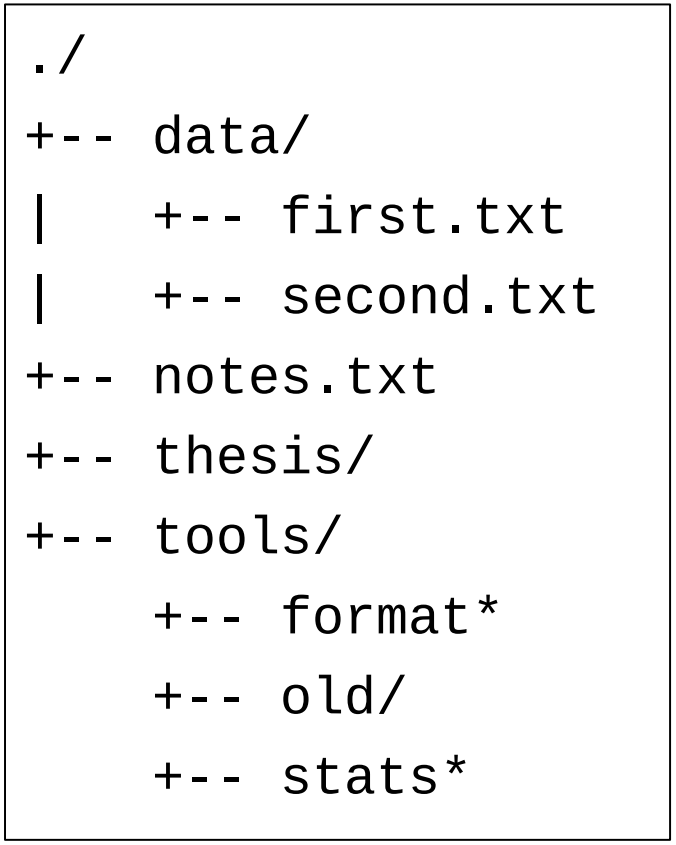
```
$ find . -type d
./
./data
./thesis
./tools
./tools/old
$
```

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -type d
./
./data
./thesis
./tools
./tools/old
$ find . -type f
./data/first.txt
./data/second.txt
./notes.txt
./tools/format
./tools/stats
$
```




```
$ find . -maxdepth 1 -type f
./notes.txt
$
```



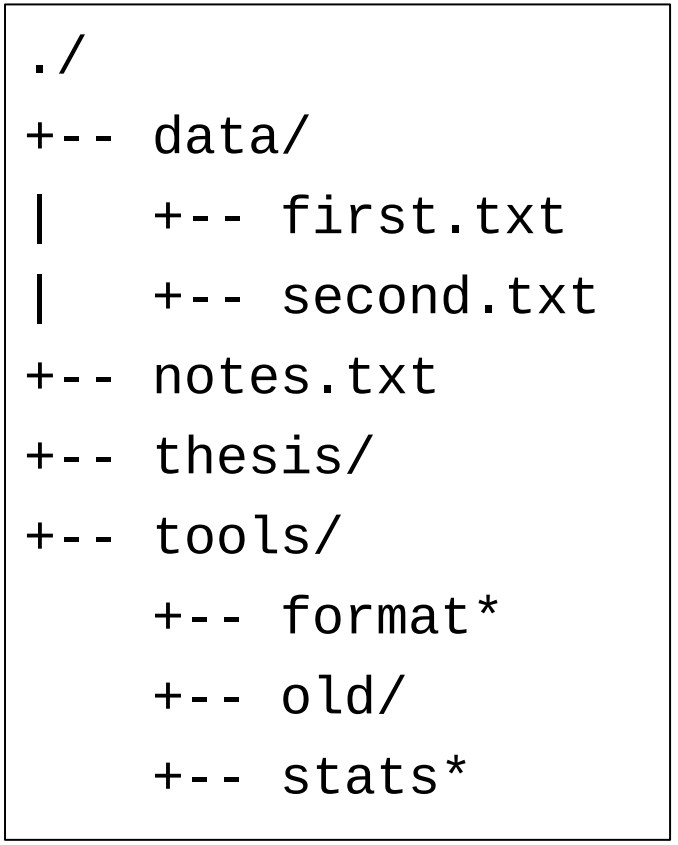

```
$ find . -maxdepth 1 -type f
./notes.txt
$ find . -mindepth 2 -type f
./data/first.txt
./data/second.txt
./tools/format
./tools/stats
$
```

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -maxdepth 1 -type f
./notes.txt
$ find . -mindepth 2 -type f
./data/first.txt
./data/second.txt
./tools/format
./tools/stats
$ find . -empty
./thesis
./tools/old
$
```

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -perm -u=x
./data
./thesis
./tools
./tools/format
./tools/old
./tools/stats
$
```




```
$ find . -perm -u=x
./data
./thesis
./tools
./tools/format
./tools/old
./tools/stats
$ find . -perm -u=x -type f
./tools/format
./tools/stats
$
```

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -name *.txt
./notes.txt
$
```

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -name *.txt
./notes.txt
$
```

* expanded by shell *before* command runs

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -name notes.txt
./notes.txt
$
```

* expanded by shell *before* command runs

This is the actual command

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -name *.txt
./notes.txt
$ find . -name '*.txt'
```

Single quotes prevent shell from expanding wildcards

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -name *.txt
./notes.txt
$ find . -name '*.txt'
```

Single quotes prevent shell from expanding wildcards

So find gets the pattern

```
./
+-- data/
|   +-- first.txt
|   +-- second.txt
+-- notes.txt
+-- thesis/
+-- tools/
    +-- format*
    +-- old/
    +-- stats*
```

```
$ find . -name *.txt
./notes.txt
$ find . -name '*.txt'
./data/first.txt
./data/second.txt
./notes.txt
$
```

The command line's power lies in *combining* tools

The command line's power lies in *combining* tools

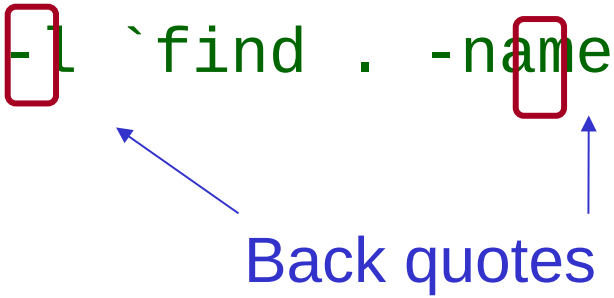
```
$ find . -name '*.txt'
./data/first.txt
./data/second.txt
./notes.txt
$
```

The command line's power lies in *combining* tools

```
$ find . -name '*.txt'
./data/first.txt
./data/second.txt
./notes.txt
$ wc -l `find . -name '*.txt'`
```

The command line's power lies in *combining* tools

```
$ find . -name '*.txt'
./data/first.txt
./data/second.txt
./notes.txt
$ wc -l `find . -name '*.txt'`
```



The command line's power lies in *combining* tools

```
$ find . -name '*.txt'
./data/first.txt
./data/second.txt
./notes.txt
$ wc -l `find . -name '*.txt'`
```

Back quotes

Replace what's inside with output from running that command

The command line's power lies in *combining* tools

```
$ find . -name '*.txt'
./data/first.txt
./data/second.txt
./notes.txt
$ wc -l `find . -name '*.txt'`
```

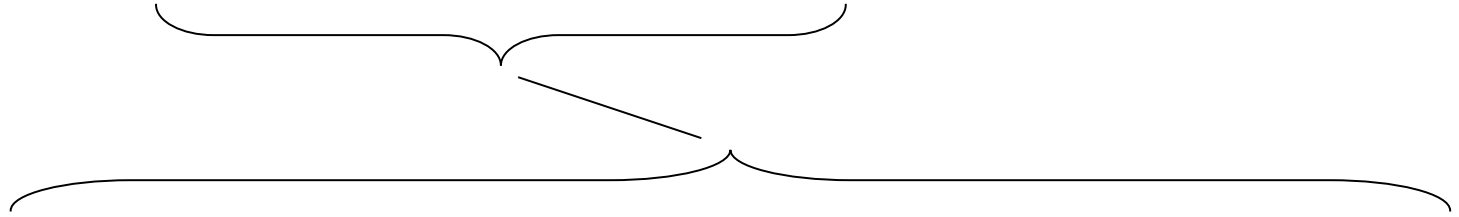
Back quotes

Replace what's inside with output from running that command

Like wildcards * and ?, but more flexible

The command line's power lies in *combining* tools

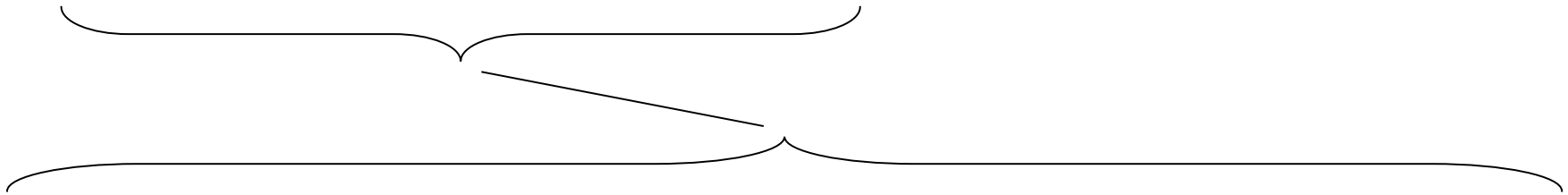
```
$ find . -name '*.txt'
./data/first.txt
./data/second.txt
./notes.txt
$ wc -l `find . -name '*.txt'`
```



```
  ./data/first.txt ./data/second.txt ./notes.txt
```

The command line's power lies in *combining* tools

```
$ find . -name '*.txt'
./data/first.txt
./data/second.txt
./notes.txt
$ wc -l `find . -name '*.txt'`
```



```
$ wc -l ./data/first.txt ./data/second.txt ./notes.t
```

The command line's power lies in *combining* tools

```
$ find . -name '*.txt'
./data/first.txt
./data/second.txt
./notes.txt
$ wc -l `find . -name '*.txt'`
  70  ./data/first.txt
 420  ./data/second.txt
  30  ./notes.txt
 520  total
$
```

Use `find` and `grep` together

Use `find` and `grep` together

```
$ grep FE `find . -name '*.pdb'`
./human/heme.pdb:ATOM  25  FE  1  -0.924  0.535  -0.
$
```

What if your data isn't text?

What if your data isn't text?

Images, databases, spreadsheets…

What if your data isn't text?

Images, databases, spreadsheets…

1. Teach standard tools about all these formats

What if your data isn't text?

Images, databases, spreadsheets…

1. Teach standard tools about all these formats

   Hasn't happened, and probably won't

What if your data isn't text?

Images, databases, spreadsheets…

1. Teach standard tools about all these formats

   Hasn't happened, and probably won't

2. Convert data to text (or extract text from data)

What if your data isn't text?

Images, databases, spreadsheets…

1. Teach standard tools about all these formats

   Hasn't happened, and probably won't

2. Convert data to text (or extract text from data)

   Simple things are easy

What if your data isn't text?

Images, databases, spreadsheets…

1. Teach standard tools about all these formats

   Hasn't happened, and probably won't

2. Convert data to text (or extract text from data)

   Simple things are easy

   Complex things are impossible

What if your data isn't text?

Images, databases, spreadsheets…

1. Teach standard tools about all these formats

   Hasn't happened, and probably won't

2. Convert data to text (or extract text from data)

   Simple things are easy

   Complex things are impossible

3. Use a programming language

What if your data isn't text?

Images, databases, spreadsheets…

1. Teach standard tools about all these formats

   Hasn't happened, and probably won't

2. Convert data to text (or extract text from data)

   Simple things are easy

   Complex things are impossible

3. Use a programming language

   Many have borrowed ideas from the shell

created by

Greg Wilson

August 2010