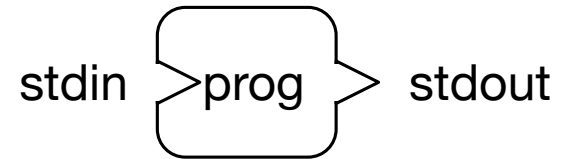


Streams/pipes/redirection in the terminal

Terence Parr
MSDS program
University of San Francisco

Streams are a unifying concept in UNIX



- Files, websites, keyboards, ... can all be accessed as streams
- Every UNIX process has:
 - a current working directory
 - standard **input** (defaults to keyboard input)
 - standard **output** (defaults to terminal output)
 - (standard **error**)

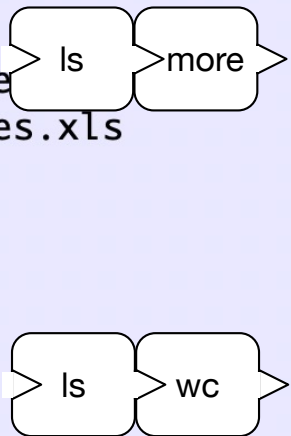
```
$ wc
501 was ok
692 is great!
      2      6     25
$
```

```
$ ls ~/github/msds692/data
AAPL.csv          TeslaIPO.html      berlitz1/
FB-AAPL-2015.csv  bbc/               berlitz1.7z
SampleSuperstoreSales.csv  bbc.7z            slate.7z
SampleSuperstoreSales.xls  bbc.zip
```

UNIX has lots of commands we can mix and match to solve problems without new code

- To combine programs, we need to send the output of one program to the input of another
- This lets us transform or simplify data in multiple steps
- The mechanism for passing the standard output of one program to the standard input of another program is called a *pipe*
- Here's an example piping the output of **ls** to the input of **more** and then **wc**:

```
$ ls ~/github/msds692/data | more
AAPL.csv
FB-AAPL-2015.csv
SampleSuperstoreSales
SampleSuperstoreSales.xls
TeslaIP0.html
bbc/
bbc.7z
bbc.zip
berlitz1/
berlitz1.7z
slate.7z
$ ls ~/github/msds692/data | wc
    11     11    143
```



Simple data science pipe example


- Sometimes it's easier to search big data files from the command line, instead of writing code
- Here's how to find all records in a CSV file associated with Backhoe bulldozers:

```
varmint:master:~/github/msds621/data $ grep Backhoe bulldozer-train.csv | wc -l  
79416
```

Deeper pipelines

- Pipe the output of **ls** to **grep** (search for string in line) and send that output to **wc**, which counts how many filenames contain “bbc”

```
$ ls ~/github/msds692/data | grep bbc  
bbc/  
bbc.7z  
bbc.zip  
$ ls ~/github/msds692/data | grep bbc | wc
```



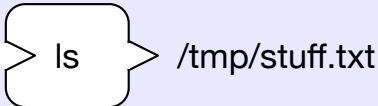
```
graph LR; A[ls] --> B[grep]; B --> C[wc];
```

The diagram illustrates the pipeline components. The first command, `ls ~/github/msds692/data | grep bbc`, lists files and filters for those containing "bbc". The second command, `ls ~/github/msds692/data | grep bbc | wc`, counts the filtered files. The output of the second command is 20, indicating that 20 files contain the string "bbc".

I/O redirection

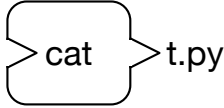
- Pipes connect process input/output
- Redirection:
 - **< file**
hooks process standard input to file
 - **> file**
hooks process standard output to file

```
$ ls ~/github/msds692/data > /tmp/stuff.txt
$ cat /tmp/stuff.txt
AAPL.csv
FB-AAPL-2015.csv
SampleSuperstoreSales.csv
SampleSuperstoreSales.xls
TeslaIPO.html
bbc/
bbc.7z
bbc.zip
berlitz1/
berlitz1.7z
slate.7z
```



A diagram showing a callout box containing the text 'ls' with an arrow pointing to the file path '/tmp/stuff.txt', indicating that the output of the 'ls' command is being redirected to that file.

```
$ cat > t.py
print("692 is great!")
$ python t.py
692 is great!
```



A diagram showing a callout box containing the text 'cat' with an arrow pointing to the file path 't.py', indicating that the output of the 'cat' command is being redirected to that file.

I'm hitting control-D (EOF) here to indicate end of input

Data science redirection example

- When developing a model or performing an analysis, it's often good to start with a small subset of the data to speed up development
- Here's how to get the first 5000 rows of data from a CSV file into another file without having to write code:

```
[varmint:~/data $ head -5000 GDP.csv > GDP-5000.csv  
[varmint:~/data $ wc -l GDP*.csv  
3137 GDP-5000.csv  
3137 GDP.csv  
6274 total
```

Redirecting standard input

- Less common but still useful
- Many commands take both commandline arguments and redirection: **sort**, **cat**, **wc** etc...

Control-D
(EOF)

```
$ cat > /tmp/stuff.txt
3
2
1
$ sort /tmp/stuff.txt
1
2
3
$ sort < /tmp/stuff.txt
1
2
3
$ cat < /tmp/stuff.txt
3
2
1
$
```


Throwing away program output

- file **/dev/null** is a special “device” that accepts input and does nothing with it
- It’s a great way to hide all of that debugging output you have in your program

```
$ ls ~/github/msds692/data > /dev/null  
$ cat /dev/null  
$
```

Appending standard output

- If you want to send the output of multiple commands to a file, you can use the append redirection operator, which looks >>

```
$ echo "501" > /tmp/stuff.txt
$ echo "692" >> /tmp/stuff.txt
$ echo "621" >> /tmp/stuff.txt
$ cat /tmp/stuff.txt
501
692
621
$
```

Summary

- pipe “|” hooks output of one process to input of another
a | b | c
- redirect program output to a file using “>”
- redirect and append program output to a file using “>>”
- open file and send contents as standard input to a program using ‘<’ operator
- Redirect both: **prog < infile > outfile**
- Redirect to first program then pipes then redirect:
a < infile | b | c | d > outfile