Streams, redirection, piping

Streams

- a data stream is an ordered sequence of bytes
  - programs can read the stream
  - programs can write to a stream

Standard streams

- standard input is a stream from which data can be read
  - on Unix-like systems, standard input is connected to the keyboard by default
- standard output is a stream to which normal output data can be written
  - on Unix-like systems, standard output is connected to the terminal by default
- standard error is a stream to which error or diagnostic output data can be written
  - on Unix-like systems, standard error is connected to the terminal by default

Standard streams

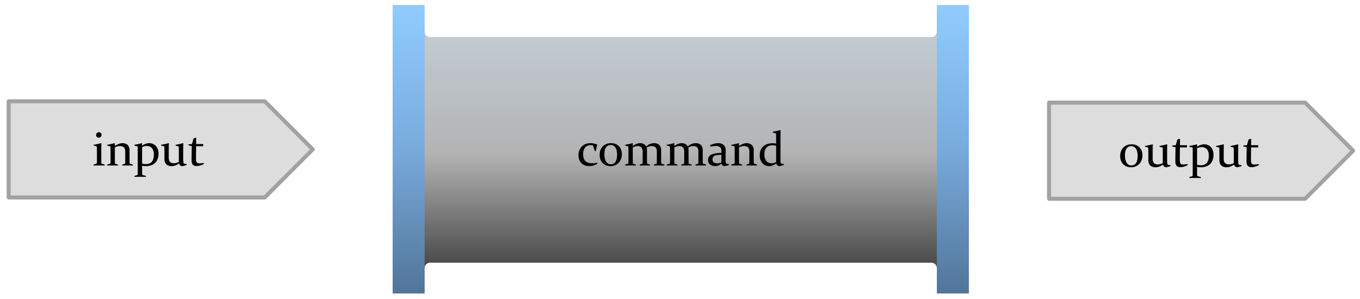
- each standard stream has a unique integer identifier called a *file descriptor*
  - **`0`** standard input
  - **`1`** standard output
  - **`2`** standard error

Redirection

- many commands

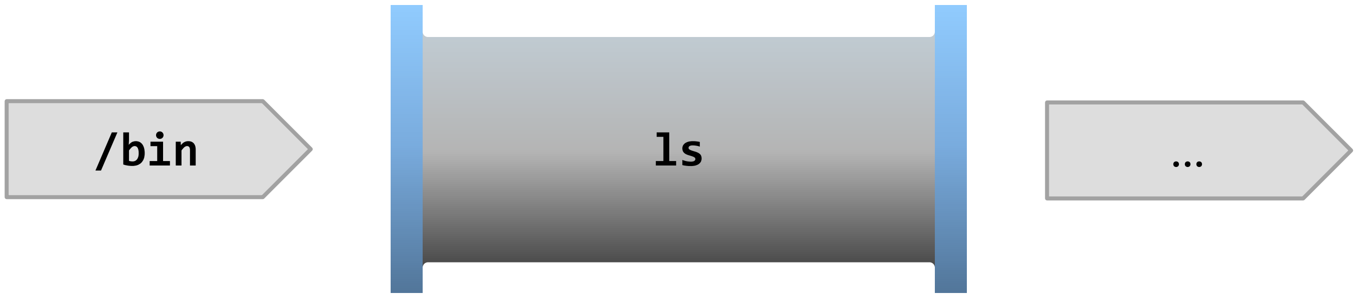
Pipelines

- commands in Linux are like segments of pipe
  - input flows in one end
  - output flows out of the other end



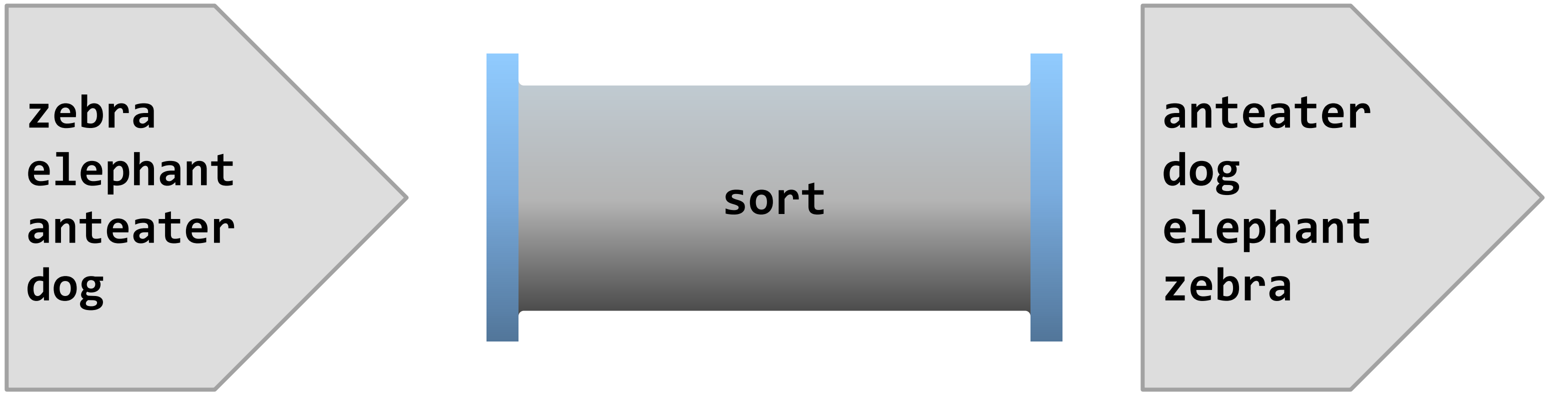
Pipelines

- input can take the form of a command line argument
  - e.g., **`ls /bin`**



Pipelines

- input can come from standard input for some commands
  - CTRL-d signifies the end of input
  - e.g., **`sort`**

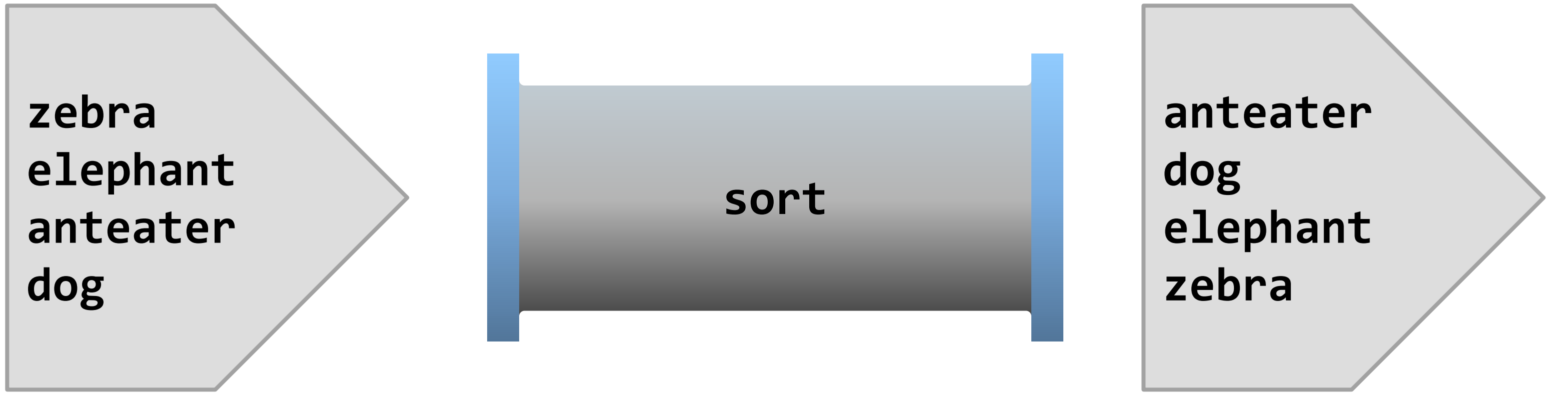


`cowsay` will also take its input from standard input

```
cowsay
```

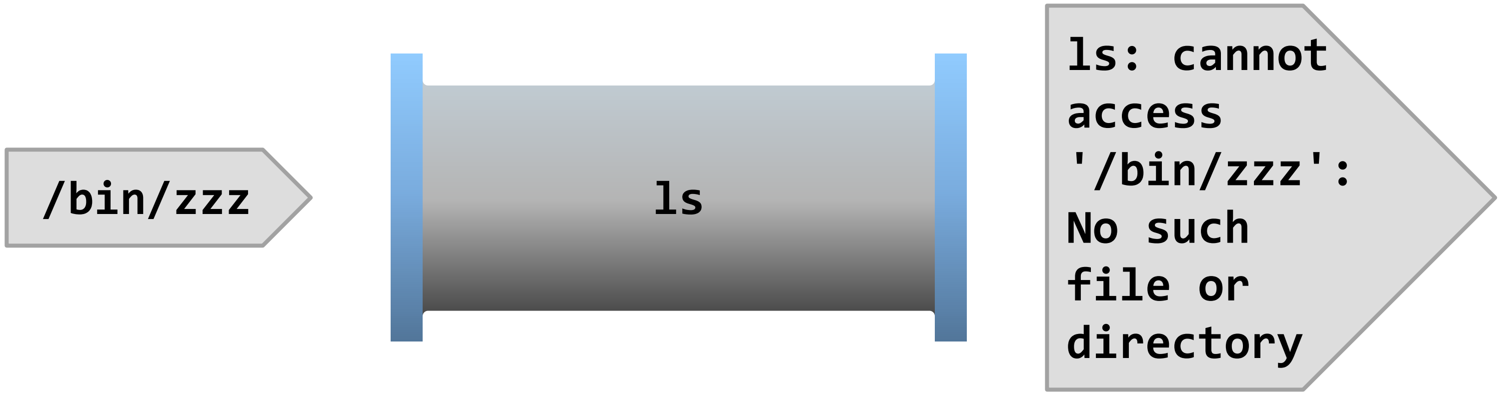
Pipelines

- command output usually goes to standard output



Pipelines

- command output usually goes to standard error if the command encountered some kind of error
  - e.g., **`ls /bin/zzz`**



```
ls: cannot
access
'/bin/zzz':
No such
file or
directory
```

Redirection

- the user can change where a command gets its input from or sends its output to via *redirection*
- redirecting standard output or standard error allows the user to send the output of a command to a file
- redirecting standard input allows the user to send the contents of a file as input to a command

Redirecting standard output

- to redirect standard output, place

  `1>` *`output_filename`*

  after the command and its arguments where *`output_filename`* is the name of the file that you want to write the output to
  - the **`1`** is the file descriptor
  - creates the file if necessary, or overwrites the contents of the file if it already exists

To list all of the files except `.` and `..` in the current directory and save the output in **`files.txt`** do:

```
ls -A 1> files.txt
```

Redirecting standard output is more common than redirecting standard error, so the file descriptor is optional:

```
ls -A > files.txt
```

Any command that sends its output to standard output can have its output redirected to a file:

```
cowsay -f dragon "Mmm, crunchy knight" > moo.txt
```

Instead of overwriting the output file, appending to the output file can be done using `>>`:

```
ls -A >> files.txt
```

Redirecting standard error requires the use of **`2>`**:

```
ls /bin/zzz 2> error.txt
```

Both standard output and standard error streams can be redirected for the same command:

```
ls /bin /bin/zzz > files.txt 2> error.txt
```

Redirection standard input

- to redirect standard input so that it reads the contents of a file (instead of reading the keyboard) write:

  `0<` *`input_filename`*

  after the command and its arguments where *`input_filename`* is the name of the file that you want to use as input to the command

  - the **`0`** is the file descriptor

Most commands already accept files as inputs

- but you can still use redirection if you want
  - e.g., assume that you have a file named **`unsorted.txt`**

```
sort 0< unsorted.txt
```

The file descriptor is optional for input redirection:

```
sort < unsorted.txt
```

You can redirect both the input and output:

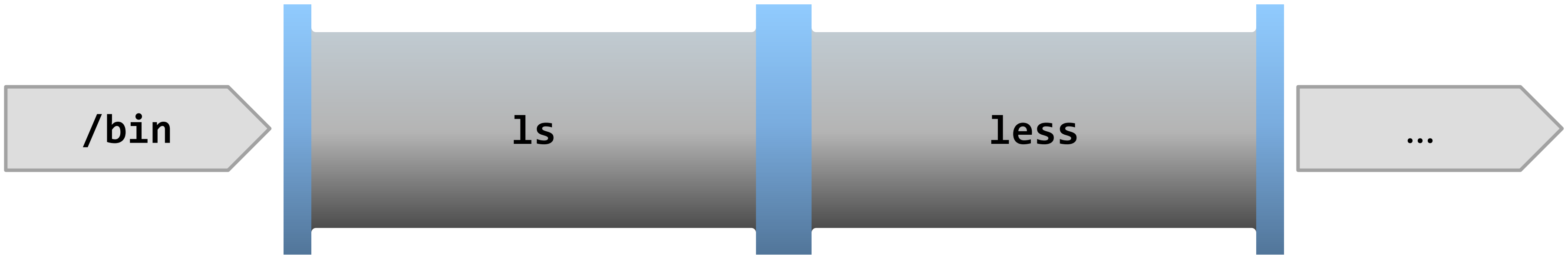
```
sort < unsorted.txt > sorted.txt
```

Pipelines

- commands accept inputs and have outputs
- can you connect the output of a command to the input of a second command?
  - yes!

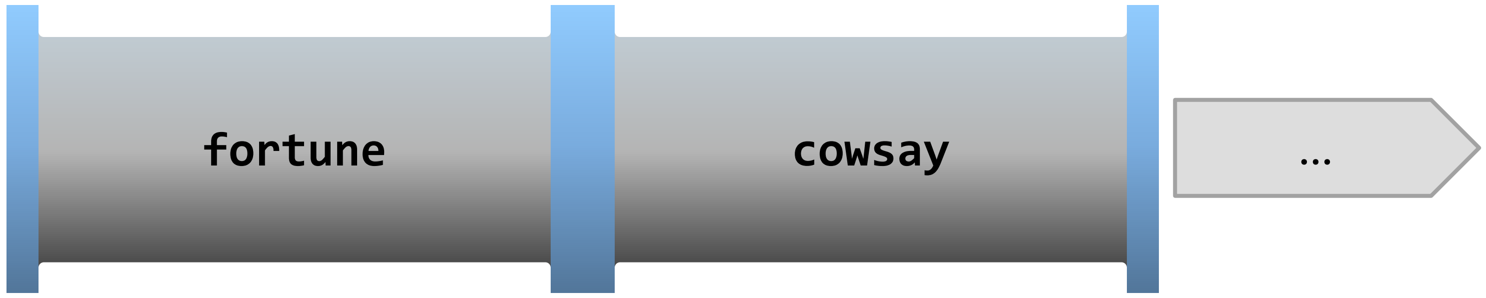
Pipelines

- use **|** to connect the output of one command to the input of the following command
  - e.g., **`ls /bin | less`**



Pipelines

- use **|** to connect the output of one command to the input of the following command
  - e.g., **`fortune | cowsay`**



You can connect as many commands as you require:

```
ls /bin /usr/bin | sort | less
```

How many files in total are in the two directories?

```
ls /bin /usr/bin | sort | wc -l
```

How many unique files (unique filenames) in total are in the two directories?

```
ls /bin /usr/bin | sort | uniq | wc -l
```

How many duplicated filenames in total are in the two directories?

```
ls /bin /usr/bin | sort | uniq -d | wc -l
```