Advanced Shell

Helen Cook

CBioVikings Viking Thursdays

Nov 3, 2016

git clone http://github.com/bitmask/type-even-less

Ask loudly if you have questions

Shout loudly if you have a better solution

~$

We are not
here because
we like to type

The command line is customizable

The command line gives you more power

Laziness

Impatience

Hubris


LG

Last time…

git clone

http://github.com/bitmask/type-less

This time…

git clone

http://github.com/bitmask/type-even-less

Install git

http://desktop.github.com/

or

http://git-scm.com/downloads

Mac OS X

homebrew
package manager

http://brew.sh/

Mac OS X

colordiff
coreutils
git
grep

brew tap homebrew/dups

brew install

moreutils
parallel
pymol
tmux
wget

Mac OS X

install iterm2

http://iterm2.com

Mac OS X

open .

A choice of shells

```
Path: senator-bedfellow.mit.edu!bloom-beacon.m
From: Tom Christiansen <tchrist@mox.perl.com>
Newsgroups: comp.unix.shell,comp.unix.question
Subject: Csh Programming Considered Harmful
Followup-To: comp.unix.shell
Date: 6 Oct 1996 14:03:18 GMT
Organization: Perl Training and Consulting
Lines: 558
Approved: news-answers-request@MIT.Edu
Expires: Sun, 1 Dec 1996 12:00:00 GMT
Message-ID: <538e76$8uq$1@csnews.cs.colorado.e
NNTP-Posting-Host: perl.com
Originator: tchrist@mox.perl.com
Xref: senator-bedfellow.mit.edu comp.unix.shel
```

“I am continually shocked and dismayed to see people … using the csh.”

I like **zsh**

The default nearly everywhere is **bash** (it is fine)

Everything in this presentation works with both zsh and bash

Variables

$PATH

How do you see what's on your path?

echo $PATH

How do you add a dir to your path?

PATH=/new/dir:$PATH

PATH=/new/dir:$PATH

Separator is a colon

How do you see all the variables that are set?

env

Aside: run perl in a script on different machines with a single #! line

#!/usr/bin/env perl

How do you set a new variable?

On the command line:

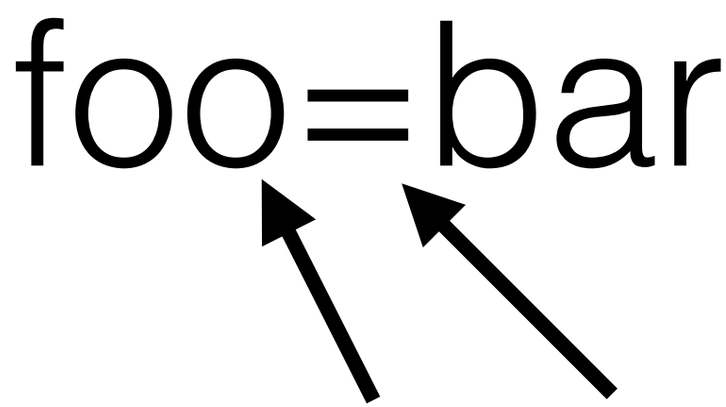
```
foo=bar
```

In a script:

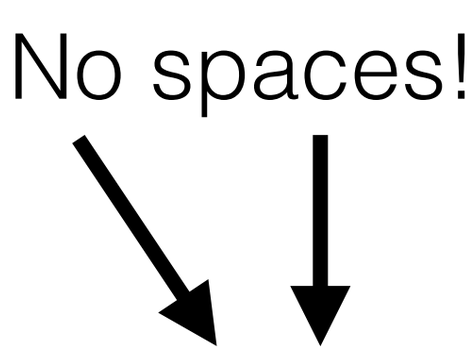
```
export foo="bar"
```

available to sub processes

On the command line:

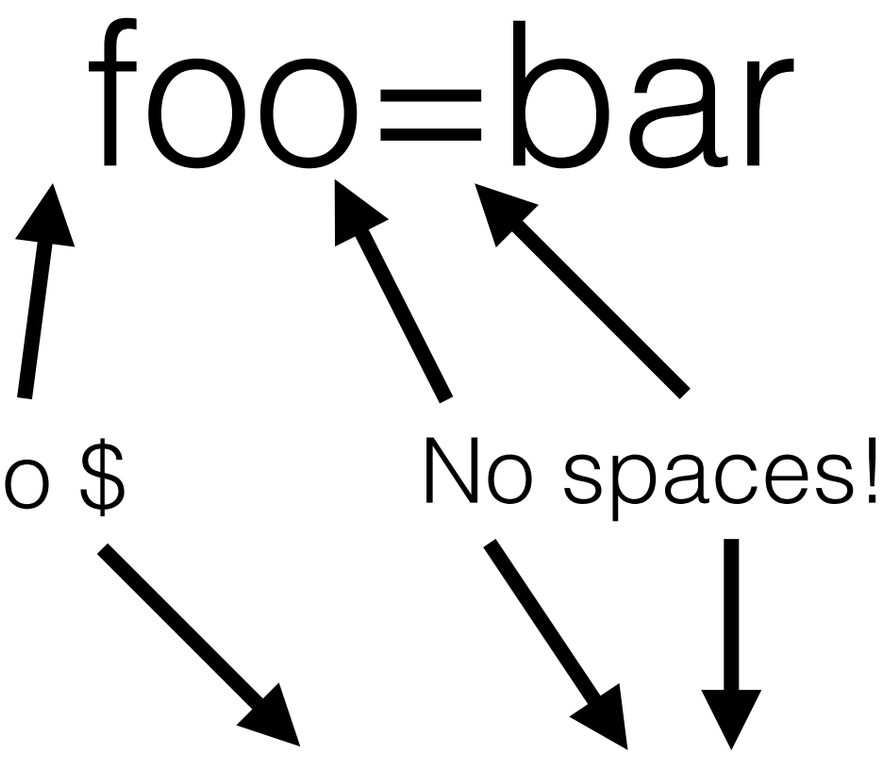


In a script:




```
export foo="bar"
```

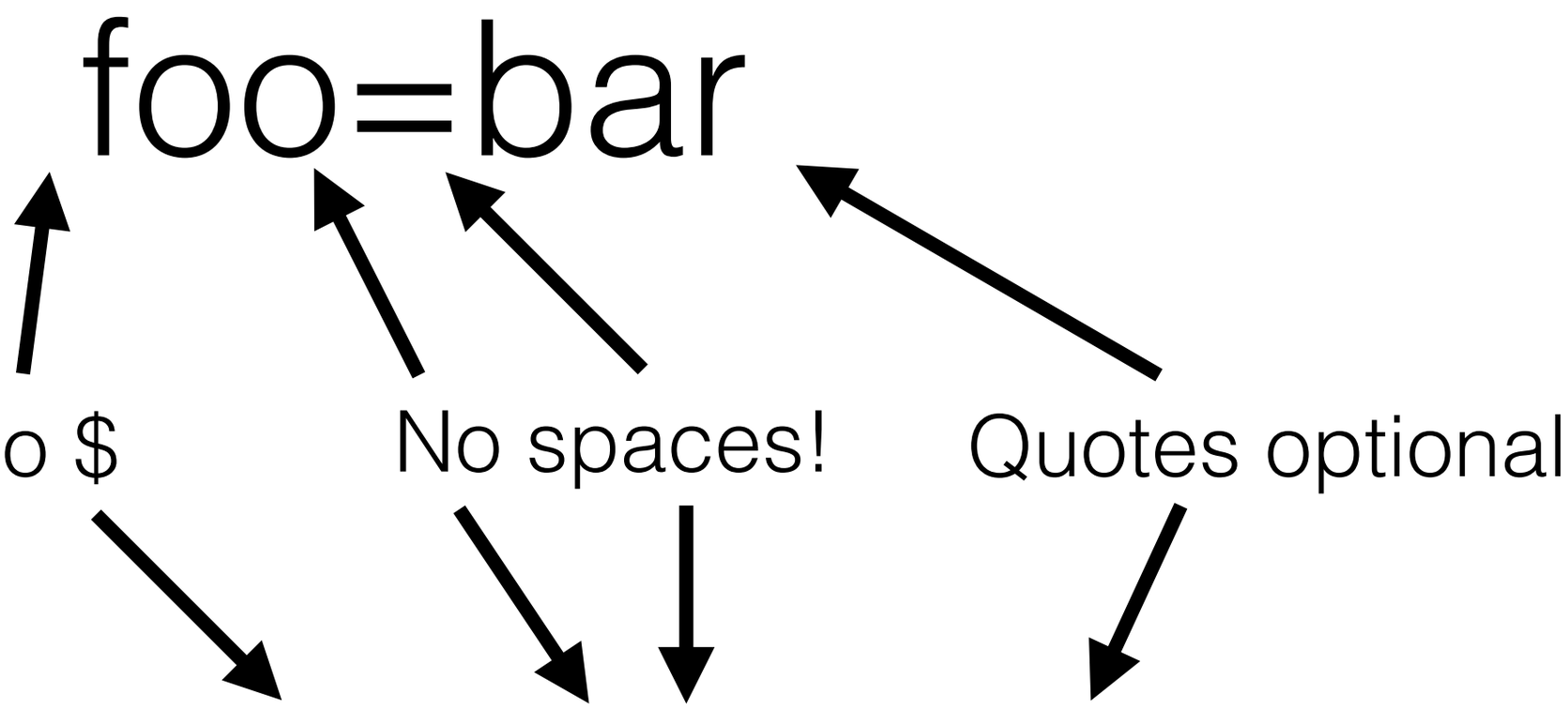
On the command line:



In a script:

```
export foo="bar"
```

On the command line:



In a script:

```
export foo="bar"
```

csh

set variable=value

setenv variable value

How do you use a variable?

$foo

exercises/dir

Exercise

Create a variable for this directory name and then use the variable to navigate to it

```
prot=~/type-even-less/exercises/dir
cd $prot

for i in $prot/*.fa; do …
```

exercises/dir

Exercise

Create a variable containing the **name of the current directory** and then use the variable to navigate.

```
export prot=.
cd $prot
```

pwd

prot=$(pwd)

Dynamic variables (subshells)

```
now=$(date)
```

```
now=$(date)
echo $now
```

```
now=$(date)
echo $now
Sun Oct 30 11:49:51 CET 2016
```

```
before=$(date) && \\
command && \\
after = $(date)

echo “$before \t $after”
```

Aside: chaining commands together

cmd1; cmd2

Aside: chaining commands together

cmd1; cmd2

cmd2 is **always** executed

Aside: chaining commands together

cmd1 && cmd2

Aside: chaining commands together

cmd1 **&&** cmd2

cmd2 is executed only if cmd1 **succeeds**

Aside: chaining commands together

cmd1 || cmd2

Aside: chaining commands together

cmd1 || cmd2

cmd2 is executed only if cmd1 **fails**

exercises/dir

Exercise

For each species in dir/*.fa
write the number of proteins
into output/*.out

```
basename prot/Q65388.fa .fa
```

```
Q65388
```

```
BN=$(basename foo.fa .fa)
```

```
for i in proteins/*.fa;
  do BN=$(basename $i .fa);
  wc -l $i > output/$BN.out;
done
```

```
for i in dir/*.fa;
  do BN=$(basename $i .fa);
  lines=$(cat $i | wc -l);
  echo $(($lines/2)) > output/$BN.out;
done
```

$(()) evaluates math

$() launches a subshell,
and will put the output
directly on the command line

Do you know another way to do this?

BN=`basename foo.txt .txt`

Be wary of using backticks

- they do not nest
- quoting is a pain

… but are one less character to type

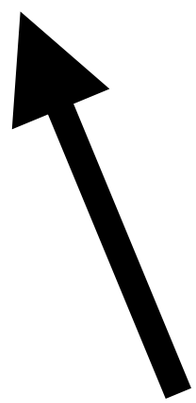
$(command)

```
var=$(command)
```

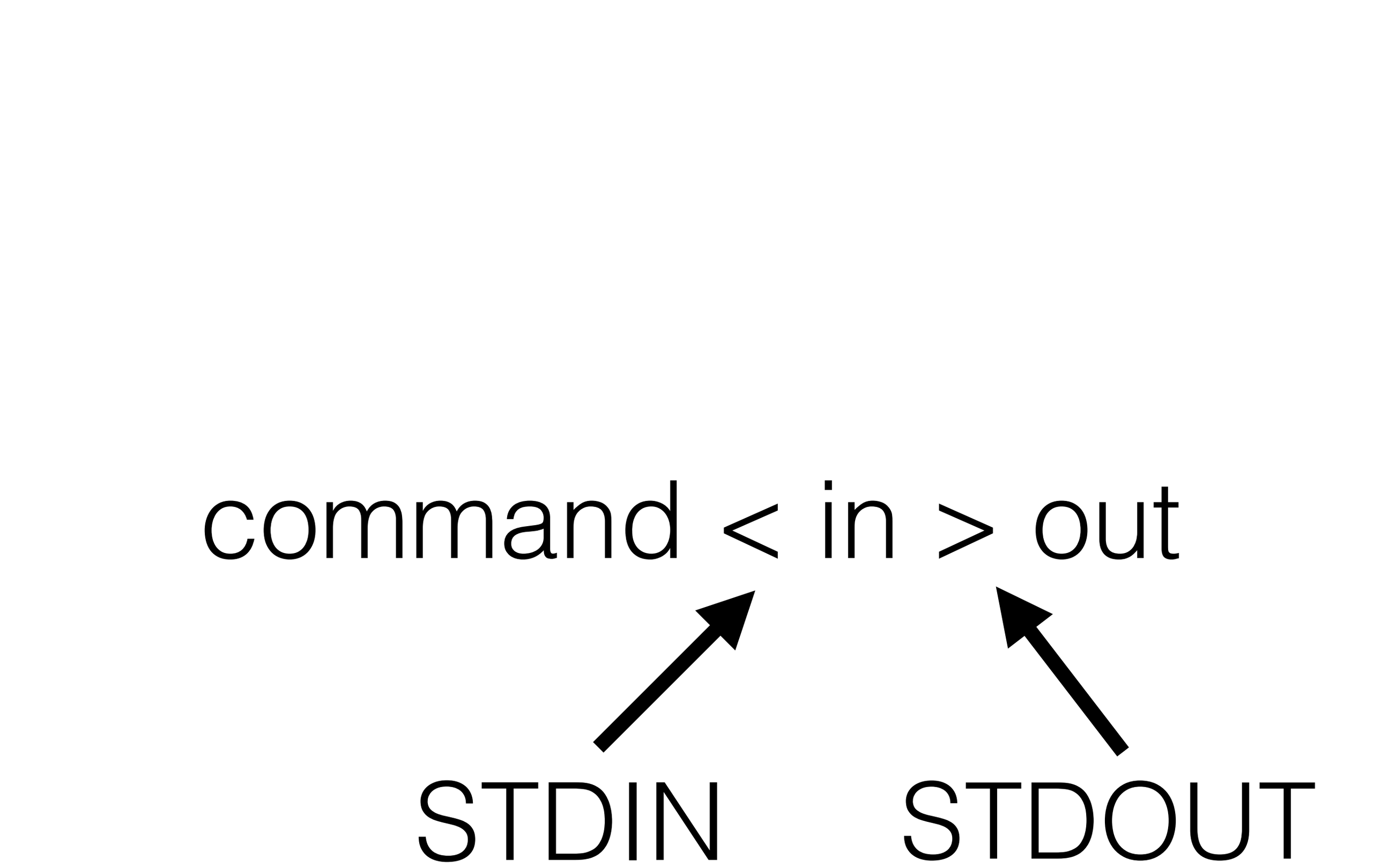
Streams
(a sneaky introduction to process substitution)

command < in > out

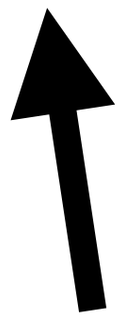
command < in >> out



append

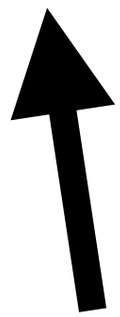

command < in > out
STDIN
STDOUT

command > out 2> err



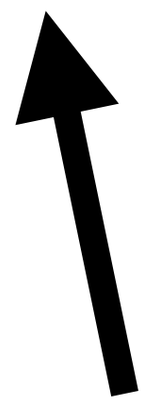
STDERR

command > out 2> err



STDERR

command &> both



both STDOUT
and STDERR

exercises/comb

Exercise

Combine two files by id

Both contain the id in the first column and different data in subsequent columns

First column of A and C contain the id that we will join on

id1 A rest of line 1 C rest of line 2
id2 A rest of line 2 C rest of line 1

exercises/comb

Hint: join

exercises/comb

join A C > AC

join input **must be sorted**
on the join column

```
sort A > A.sorted
sort C > C.sorted
join A.sorted C.sorted > AC
```

exercises/comb

join -1 1 -2 1 -o 0,1.2,1.3,2.2 A C > AC

(if A and C are sorted)

How can you do this without writing the temporary files?

exercises/comb

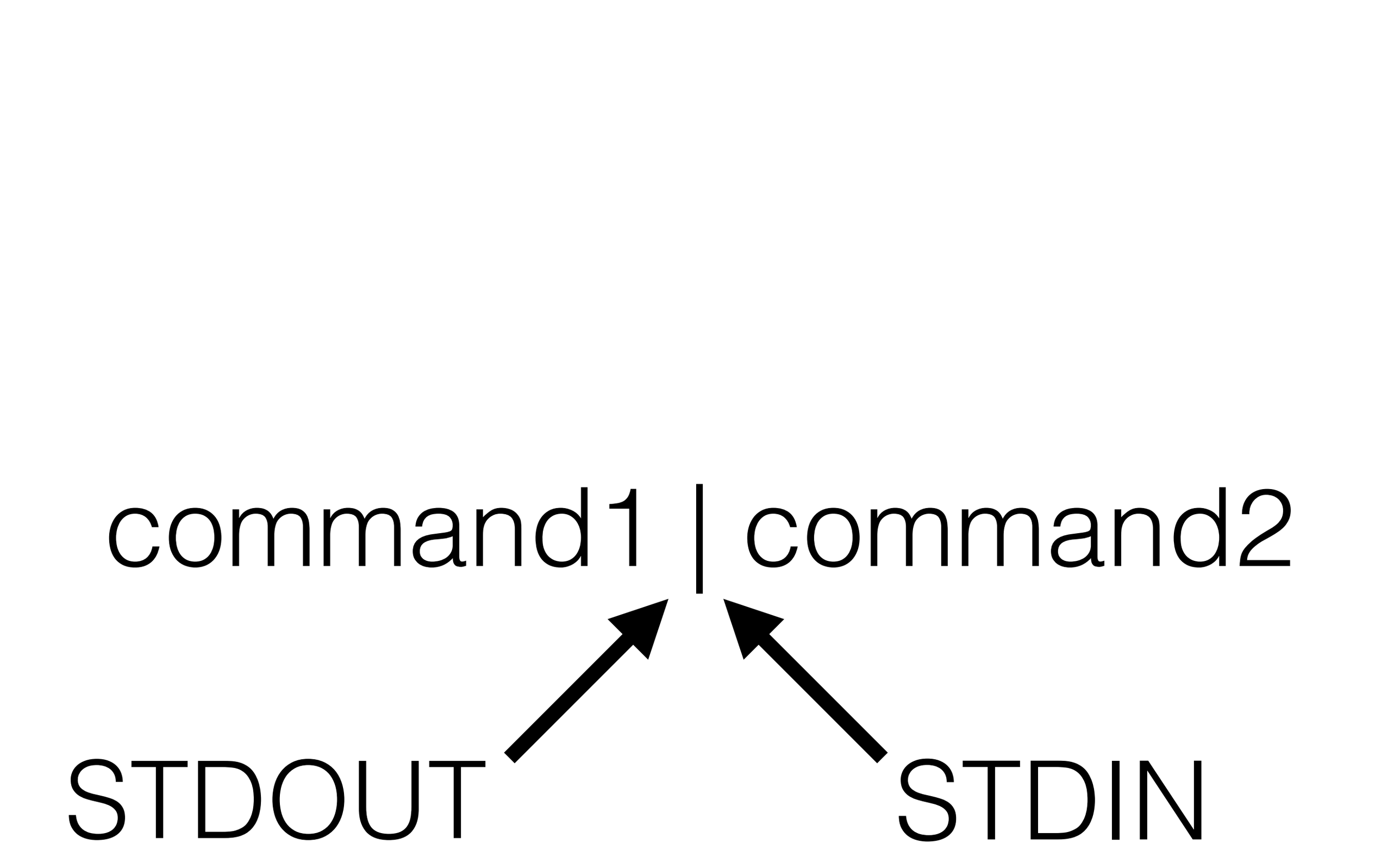
join <(sort A) <(sort C) > AC

<() puts the output into a temporary file

Pipes

command > file

command1 | command2


command1 | command2
STDOUT
STDIN

exercises/comb

Exercise

Rewrite the following using <()

```
grep "^Q" A | cut -f 2,3
```

Look for yeast mitochondrial genes, get their data

```
cut -f 2,3 <(grep "^Q" A)
```

exercises/dir

Exercise

Compare the .fa files in dir with the manifest

There is one extra file in the directory — which?

```
diff <(ls *.fa|sed "s/.fa//") manifest
```

exercises/dir

Exercise

Get the taxids out of species for every species name in interesting_species

for iterates over every word, not every line

```
for i in $(cat i_s|tr " " "_");
   do grep "$(echo $i|tr "_" " ")" species;
done
```

```
for i in "$(cat i_s)";
    do grep $i species;
done
```

or

```
IFS=$'\n'
for i in $(cat i_s);
    do grep $i species;
done
```

set in the included .zshrc file

Recap

variable=value
$variable

$(command) — put the results right here

<(command) — put the results into a temp file

$((math))

command > out
command >> append
command < in
command 2> err
command &> both

command1 | command2