Practical Computing for Scientists

Armin Sobhani
CSCI 2000U
UOIT – Fall 2015

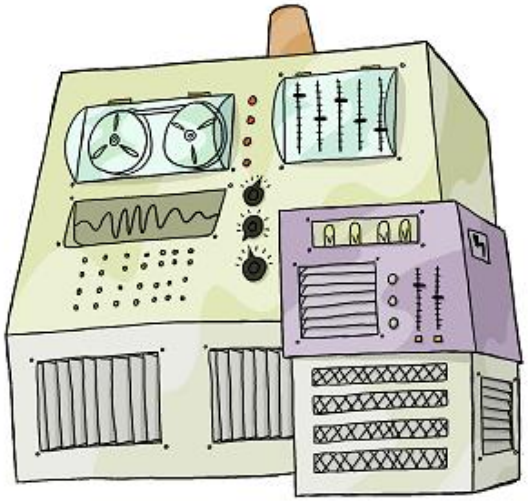
UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

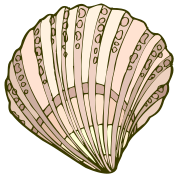
The Unix Shell

Creating and Deleting

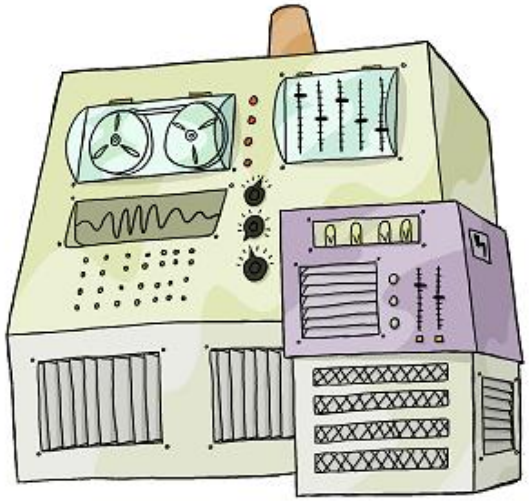
Created by Greg Wilson

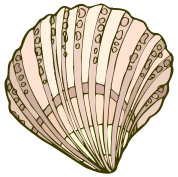
UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY





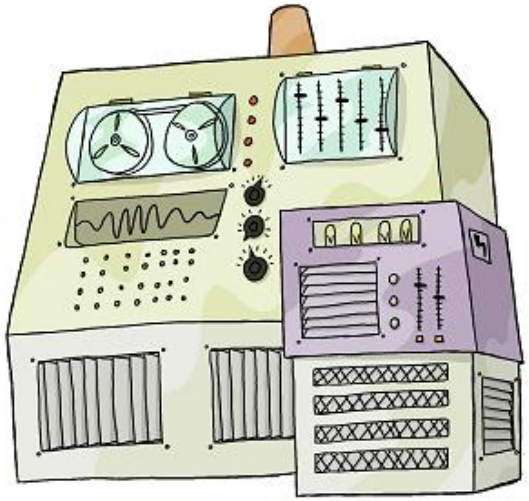
shell

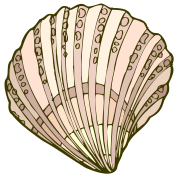




shell

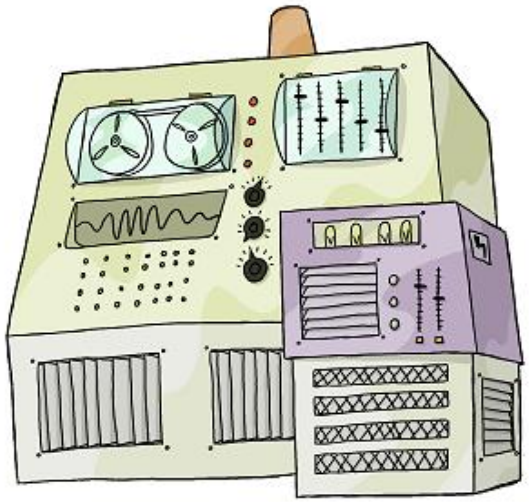
`pwd`	print working directory
`cd`	change working directory
`ls`	listing
`.`	current directory
`..`	parent directory





shell

`pwd`	print working directory
`cd`	change working directory
`ls`	listing
`.`	current directory
`..`	parent directory



But how do we create things in the first place?

```
$ pwd
/users/nelle
$
```

```
$ pwd
/users/nelle
$ ls -F
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
$
```

```
$ pwd
/users/nelle
$ ls -F
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
$ mkdir tmp
```

```
$ pwd
/users/nelle
$ ls -F
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
$ mkdir tmp
```

make directory

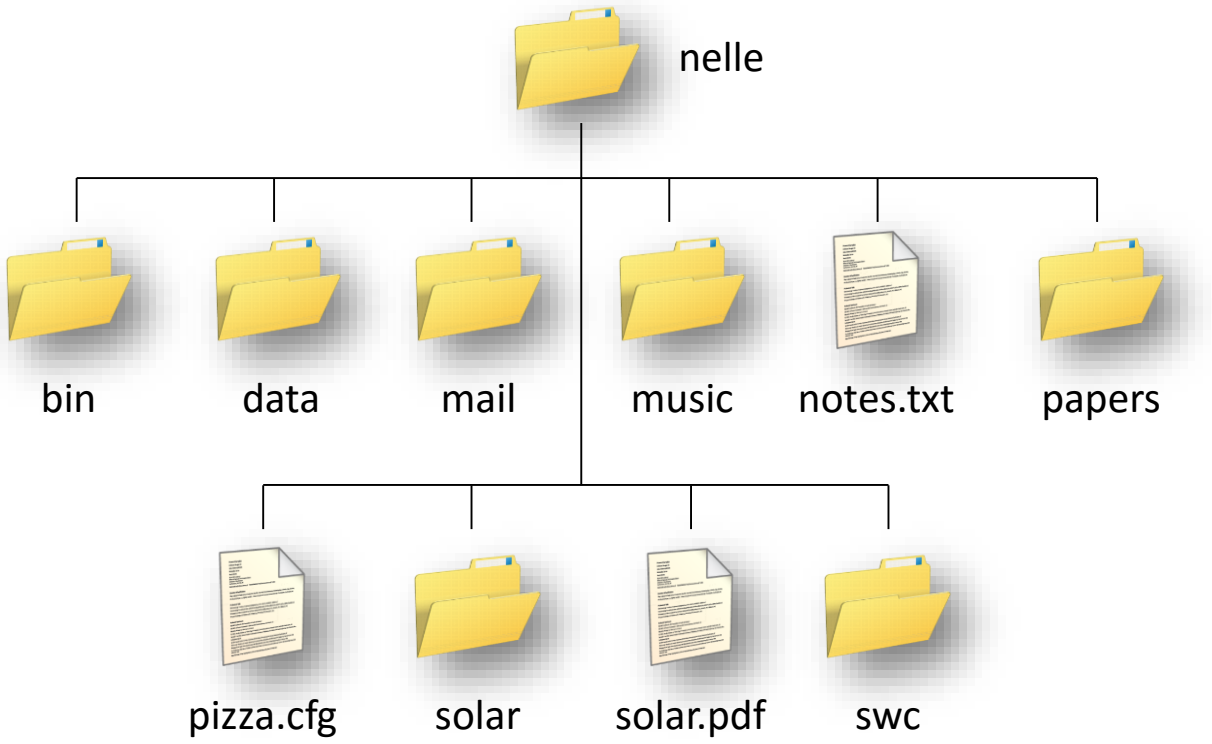
```
$ pwd
/users/nelle
$ ls -F
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
$ mkdir tmp
```

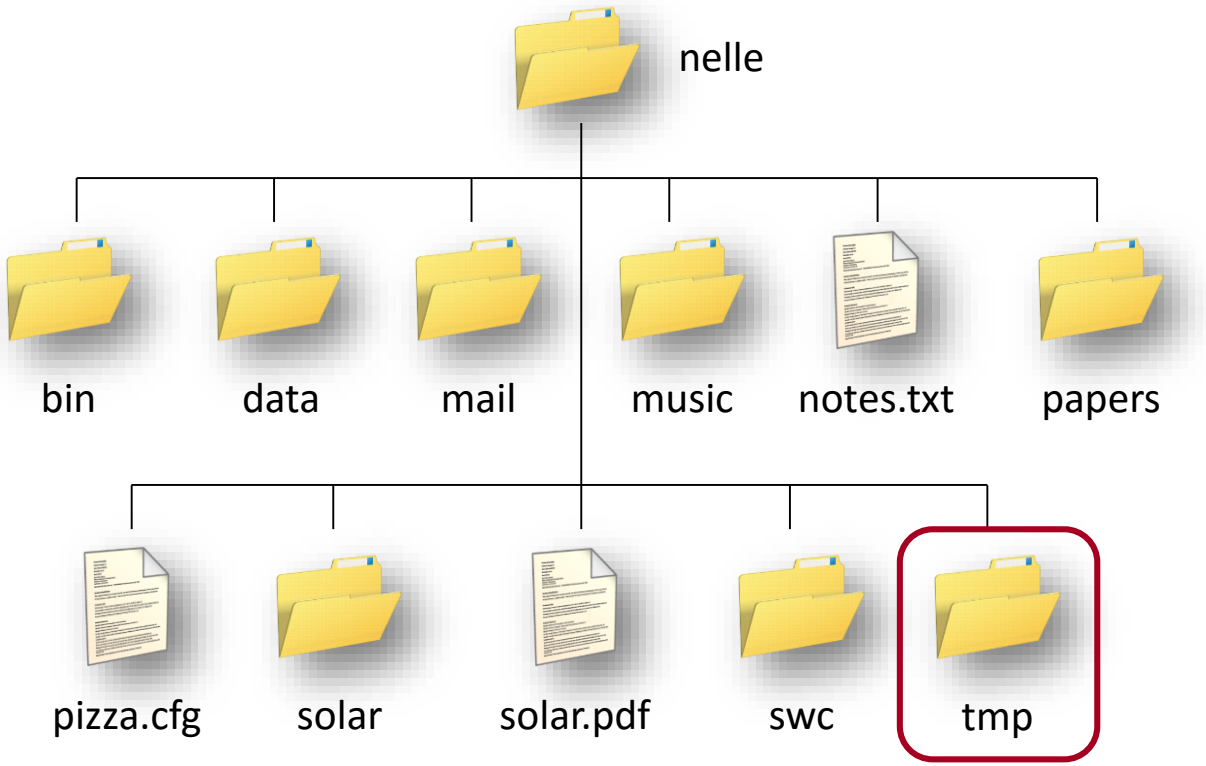
make directory

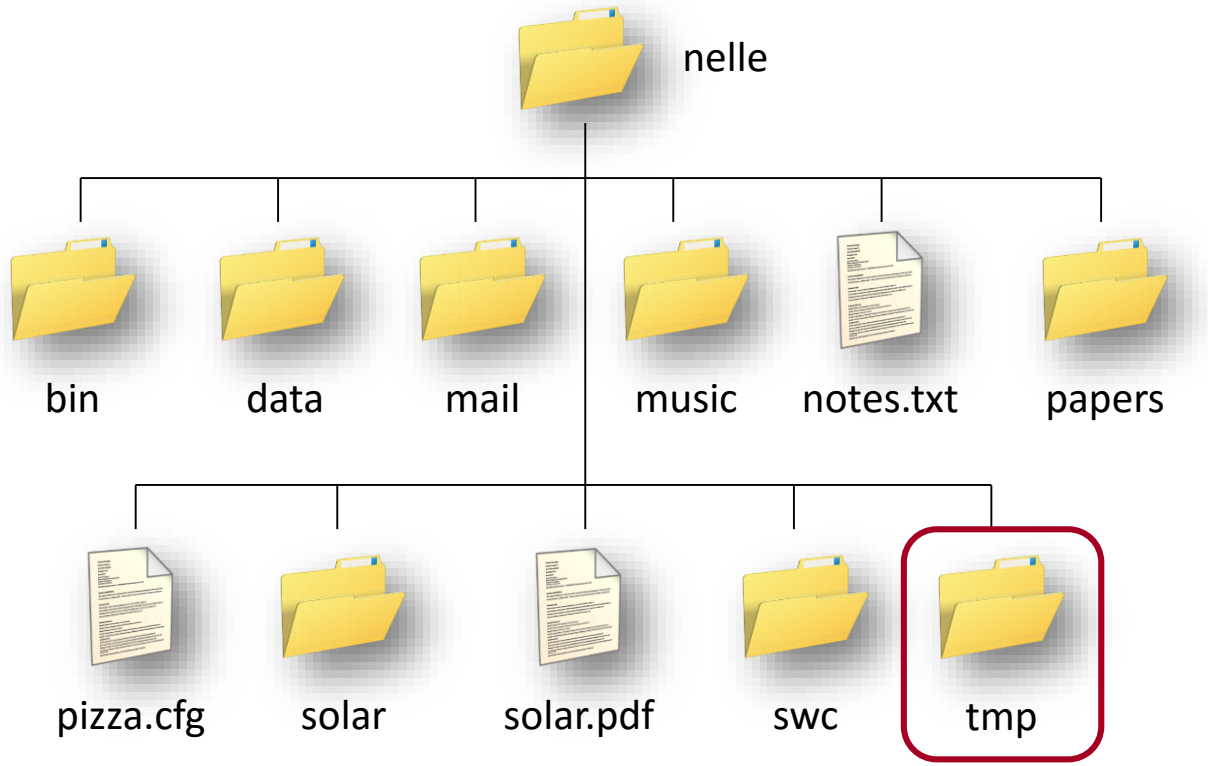
a relative path, so the new directory is made below the current one

```
$ pwd
/users/nelle
$ ls -F
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
$ mkdir tmp
$ ls -F
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/      tmp/
$
```

```
$ pwd
/users/nelle
$ ls -F
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
$ mkdir tmp
$ ls -F
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/      tmp/
$
```


nelle
bin
data
mail
music
notes.txt
papers
pizza.cfg
solar
solar.pdf
swc


nelle
bin
data
mail
music
notes.txt
papers
pizza.cfg
solar
solar.pdf
swc
tmp



nothing below it yet

```
$ pwd
/users/nelle
$
```

```
$ pwd
/users/nelle
$ ls tmp
$
```

```
$ pwd
/users/nelle
$ ls tmp
$
```

no output

```
$ pwd
/users/nelle
$ ls tmp
$ ls -a tmp
.              ..
$
```

```
$ pwd
/users/nelle
$ ls tmp
$ ls -a tmp
.              ..
```

/users/nelle/tmp

```
$ pwd
/users/nelle
$ ls tmp
$ ls -a tmp
.                ..
```

/users/nelle

```
$ cd tmp
$ nano junk
```

```
$ cd tmp
$ nano junk
```

a text editor only a programmer could love

```
$ cd tmp
$ nano junk
```

a text editor only a programmer could love

really do mean "text"...

```
$ cd tmp
$ nano junk
```

```
GNU nano 2.0.7                          File: junk



                                      [ New File ]
^G Get Help       ^O WriteOut       ^R Read File      ^Y Prev Page      ^K Cut Text       ^C Cur Pos
^X Exit           ^J Justify        ^W Where Is       ^V Next Page      ^U UnCut Text     ^T To Spell
```

```
$ cd tmp
$ nano junk
```

That's your cursor

```
GNU nano 2.0.7                          File: junk

_

                                      [ New File ]
^G Get Help       ^O WriteOut       ^R Read File      ^Y Prev Page      ^K Cut Text       ^C Cur Pos
^X Exit           ^J Justify        ^W Where Is       ^V Next Page      ^U UnCut Text     ^T To Spell
```

```
$ cd tmp
$ nano junk
```

```
GNU nano 2.0.7                              File: junk

Make everything as simple as possible,
but no simpler.␣

                                          [ New File ]
^G Get Help       ^O WriteOut       ^R Read File      ^Y Prev Page      ^K Cut Text       ^C Cur Pos
^X Exit           ^J Justify        ^W Where Is       ^V Next Page      ^U UnCut Text     ^T To Spell
```

```
$ cd tmp
$ nano junk
```

```
GNU nano 2.0.7                              File: junk

Make everything as simple as possible,
but no simpler.

                                            [ New File ]
^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

^O means "Control + O" (to save changes)

```
$ cd tmp
$ nano junk
```

```
GNU nano 2.0.7                              File: junk

Make everything as simple as possible,
but no simpler.␣




                                          [ New File ]
^G Get Help       ^O WriteOut       ^R Read File      ^Y Prev Page      ^K Cut Text       ^C Cur Pos
^X Exit           ^J Justify        ^W Where Is       ^V Next Page      ^U UnCut Text     ^T To Spell
```

^X to exit back to the shell

```
$ cd tmp
$ nano junk
$
```

nano doesn't leave any output on the screen after it exits

```
$ cd tmp
$ nano junk
$ ls
junk
$
```

but it has created the file

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$
```

use `-s` to show sizes

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$
```

use `-s` to show sizes reported in disk blocks

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$
```

use `-s` to show sizes
reported in disk blocks
a less helpful default
may have been possible…

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$
```

use `-h` for human-friendly output

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$
```

use `-h` for human-friendly output

number of bytes

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$
```

use `-h` for human-friendly output

number of bytes

rounded up because computer stores things on disk using blocks of 512 bytes

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$ rm junk
$
```

remove (delete) file

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$ rm junk
$
```

remove (delete) file

there is no (easy) un-delete!

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$ rm junk
$ ls
$
```

← check that it's gone

```
$ pwd
/users/nelle/tmp
$ nano junk
$ ls
junk
$
```

```
$ pwd
/users/nelle/tmp
$ nano junk
$ ls
junk
$ cd ..
$
```

change working directory to `/users/nelle`

```
$ pwd
/users/nelle/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove 'tmp': Is a directory
$
```

`rm` only works on files

```
$ pwd
/users/nelle/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove 'tmp': Is a directory
$ rmdir tmp
```

use `rmdir` to remove directories

```
$ pwd
/users/nelle/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove 'tmp': Is a directory
$ rmdir tmp
rmdir: failed to remove 'tmp': Directory not empty
$
```

but it only works when the directory is empty

```
$ pwd
/users/nelle/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove 'tmp': Is a directory
$ rmdir tmp
rmdir: failed to remove 'tmp': Directory not empty
$
```

but it only works when the directory is empty (safety feature)

```
$ pwd
/users/nelle/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove 'tmp': Is a directory
$ rmdir tmp
rmdir: failed to remove 'tmp': Directory not empty
$ rm tmp/junk
$
```

so get rid of the directory's contents...

```
$ pwd
/users/nelle/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove 'tmp': Is a directory
$ rmdir tmp
rmdir: failed to remove 'tmp': Directory not empty
$ rm tmp/junk
$ rmdir tmp
$
```

...then get rid of the directory

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$
```

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

move a file

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

move a file (or directory)

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

move a file (or directory)
from here...

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

move a file (or directory)

from here...

...to here

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

move a file (or directory)

from here...

...to here

renames the file!

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$
```

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$
```

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$
```

← current working directory

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$
```

Move `/users/nelle/tmp/quotes.txt` to `/users/nelle/quotes.txt`

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$ ls tmp
$
```

nothing left in `tmp`

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$ ls tmp
$ ls quotes.txt
quotes.txt
```

← `quotes.txt` now in this directory

```
$ pwd
/users/nelle/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$ ls tmp
$ ls quotes.txt
quotes.txt
```

`ls` with a file or directory argument lists that file or directory

```
$ cp quotes.txt tmp/quotations.txt
$
```

copy a file

```
$ cp quotes.txt tmp/quotations.txt
$ ls quotes.txt tmp/quotations.txt
quotes.txt   tmp/quotations.txt
$
```

```
$ cp quotes.txt tmp/quotations.txt
$ ls quotes.txt tmp/quotations.txt
quotes.txt   tmp/quotations.txt
$ rm quotes.txt
$
```

```
$ cp quotes.txt tmp/quotations.txt
$ ls quotes.txt tmp/quotations.txt
quotes.txt   tmp/quotations.txt
$ rm quotes.txt
$ ls quotes.txt tmp/quotations.txt
ls: cannot access quotes.txt: No such file or directory
tmp/quotations.txt
$
```

```
$ cp quotes.txt tmp/quotations.txt
$ ls quotes.txt tmp/quotations.txt
quotes.txt   tmp/quotations.txt
$ rm quotes.txt
$ ls quotes.txt tmp/quotations.txt
ls: cannot access quotes.txt: No such file or directory
tmp/quotations.txt
$ cp tmp/quotations.txt .
$ ls quotations.txt
quotations.txt
$
```

```
$ cp quotes.txt tmp/quotations.txt
$ ls quotes.txt tmp/quotations.txt
quotes.txt   tmp/quotations.txt
$ rm quotes.txt
$ ls quotes.txt tmp/quotations.txt
ls: cannot access quotes.txt: No such file or directory
tmp/quotations.txt
$ cp tmp/quotations.txt .
$ ls quotations.txt
quotations.txt
$
```

this is a directory, so the copy has the same name as the original file

`pwd`	print working directory
`cd`	change working directory
`ls`	listing
`.`	current directory
`..`	parent directory
`mkdir`	make a directory
`nano`	text editor
`rm`	remove (delete) a file
`rmdir`	remove (delete) a directory
`mv`	move (rename) a file or directory
`cp`	copy a file

Checkpoint 3

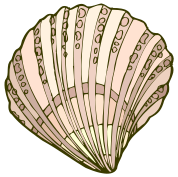
- Please complete the *What is Reality Survey Results*:
  - Blackboard > Course Content > Week 2 (Sept. 21-25) > Monday Sept. 21 > Checkpoint 3

The Unix Shell

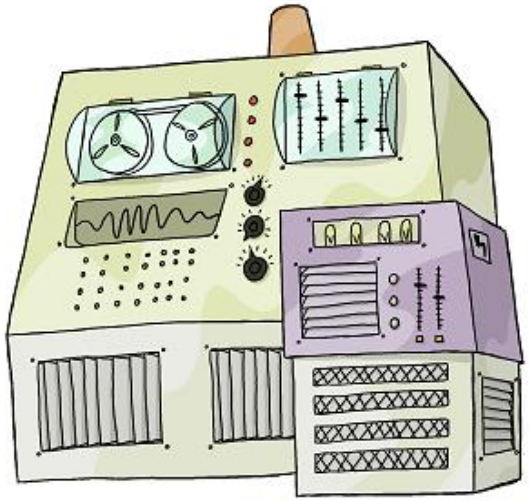
Pipes and Filters

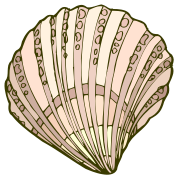
Created by Greg Wilson

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY



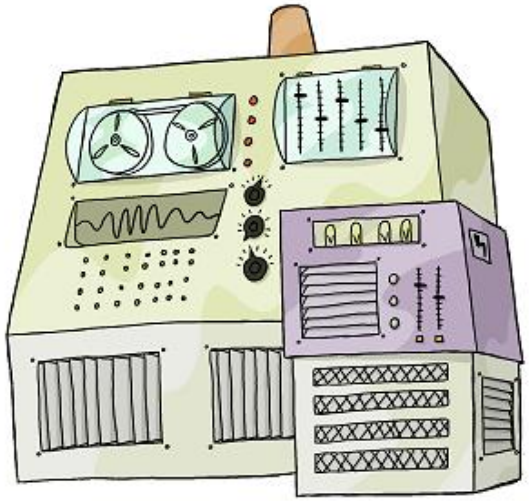
shell

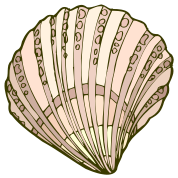




shell

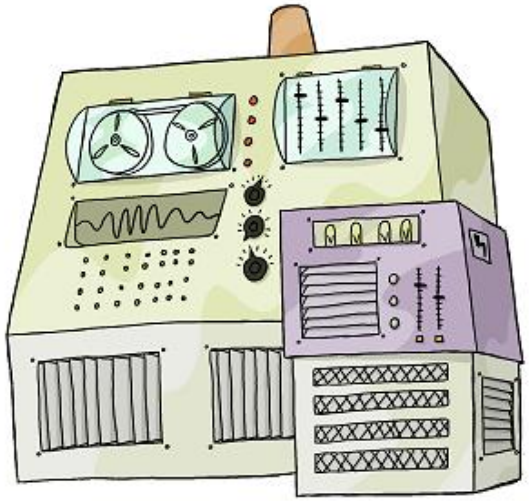
pwd	mkdir
cd	nano
ls	rm
.	rmdir
..	mv
	cp





shell

`pwd`	`mkdir`
`cd`	`nano`
`ls`	`rm`
`.`	`rmdir`
`..`	`mv`
	`cp`



More powerful

when combined

```
$ ls molecules
cubane.pdb     ethane.pdb     methane.pdb
octane.pdb     pentane.pdb    propane.pdb
$
```

```
$ ls molecules
cubane.pdb    ethane.pdb    methane.pdb
octane.pdb    pentane.pdb   propane.pdb
$ cd molecules
$
```

```
$ ls molecules
cubane.pdb    ethane.pdb    methane.pdb
octane.pdb    pentane.pdb   propane.pdb
$ cd molecules
$ wc *.pdb
```

```
$ ls molecules
cubane.pdb     ethane.pdb     methane.pdb
octane.pdb     pentane.pdb    propane.pdb
$ cd molecules
$ wc *.pdb
```

* is a *wild card*

```
$ ls molecules
cubane.pdb     ethane.pdb     methane.pdb
octane.pdb     pentane.pdb    propane.pdb
$ cd molecules
$ wc *.pdb
```

* is a *wild card*

matches zero or more characters

```
$ ls molecules
cubane.pdb     ethane.pdb     methane.pdb
octane.pdb     pentane.pdb    propane.pdb
$ cd molecules
$ wc *.pdb
```

* is a *wild card*

matches zero or more characters

so *.pdb matches all filenames

ending in .pdb

```
$ ls molecules
cubane.pdb     ethane.pdb     methane.pdb
octane.pdb     pentane.pdb    propane.pdb
$ cd molecules
$ wc *.pdb
```

```
wc cubane.pdb ethane.pdb methane.pdb octane.pdb pentane.pdb propane.pdb
```

```
$ ls molecules
cubane.pdb     ethane.pdb     methane.pdb
octane.pdb     pentane.pdb    propane.pdb
$ cd molecules
$ wc *.pdb
```

word count

```
$ ls molecules
cubane.pdb     ethane.pdb     methane.pdb
octane.pdb     pentane.pdb    propane.pdb
$ cd molecules
$ wc *.pdb
```

word count

counts lines, words, and characters in files

```
$ ls molecules
cubane.pdb     ethane.pdb     methane.pdb
octane.pdb     pentane.pdb    propane.pdb
$ cd molecules
$ wc *.pdb
  20  156 1158 cubane.pdb
  12   84  622 ethane.pdb
   9   57  422 methane.pdb
  30  246 1828 octane.pdb
  21  165 1226 pentane.pdb
  15  111  825 propane.pdb
 107  819 6081 total
$
```

```
$ wc -l *.pdb
  20  cubane.pdb
  12  ethane.pdb
   9  methane.pdb
  30  octane.pdb
  21  pentane.pdb
  15  propane.pdb
 107  total
$
```

report only lines

```
$ wc -l *.pdb
  20  cubane.pdb
  12  ethane.pdb
   9  methane.pdb
  30  octane.pdb
  21  pentane.pdb
  15  propane.pdb
 107  total
$
```

report only lines

use -w for words or

-c for characters

```
 20  cubane.pdb
 12  ethane.pdb
  9  methane.pdb
 30  octane.pdb
 21  pentane.pdb
 15  propane.pdb
107  total
```

Which file is shortest?

```
 20  cubane.pdb
 12  ethane.pdb
  9  methane.pdb
 30  octane.pdb
 21  pentane.pdb
 15  propane.pdb
107  total
```

Which file is shortest?

Easy to see when there are six…

```
 20  cubane.pdb
 12  ethane.pdb
  9  methane.pdb
 30  octane.pdb
 21  pentane.pdb
 15  propane.pdb
107  total
```

Which file is shortest?

Easy to see when there are six…

…but what if there were 6000?

```
$ wc -l *.pdb > lengths
$
```

```
$ wc -l *.pdb > lengths
$
```

redirect output to a file

```
$ wc -l *.pdb > lengths
$
```

redirect output to a file

create file if it doesn't exist

```
$ wc -l *.pdb > lengths
$
```

redirect output to a file

create file if it doesn't exist

overwrite it if it does

```
$ wc -l *.pdb > lengths
$
```

no screen output

```
$ wc -l *.pdb > lengths
$ ls lengths
lengths
$
```

```
$ wc -l *.pdb > lengths
$ ls lengths
lengths
$ cat lengths
  20  cubane.pdb
  12  ethane.pdb
   9  methane.pdb
  30  octane.pdb
  21  pentane.pdb
  15  propane.pdb
 107  total
$
```

```
$ wc -l *.pdb > lengths
$ ls lengths
lengths
$ cat lengths
  20  cubane.pdb
  12  ethane.pdb
   9  methane.pdb
  30  octane.pdb
  21  pentane.pdb
  15  propane.pdb
 107  total
$
```

concatenate files

```
$ wc -l *.pdb > lengths
$ ls lengths
lengths
$ cat lengths
  20  cubane.pdb
  12  ethane.pdb
   9  methane.pdb
  30  octane.pdb
  21  pentane.pdb
  15  propane.pdb
 107  total
$
```

concatenate files

in this case, only one

so file contents printed to screen

```
$ sort lengths
  9  methane.pdb
 12  ethane.pdb
 15  propane.pdb
 20  cubane.pdb
 21  pentane.pdb
 30  octane.pdb
107  total
$
```

```
$ sort lengths > sorted-lengths
$
```

```
$ sort lengths > sorted-lengths
$ head -1 sorted-lengths
  9  methane.pdb
$
```

```
$ sort lengths > sorted-lengths
$ head -1 sorted-lengths
  9  methane.pdb
$
```

get the first line of the file

```
$ sort lengths > sorted-lengths
$ head -1 sorted-lengths
  9  methane.pdb
$
```

get the first line of the file
this must be the PDB file
with the fewest lines,
since sorted-lengths holds
files and line counts in
order from least to greatest

```
$ sort lengths > sorted-lengths
$ head -1 sorted-lengths
  9  methane.pdb
$
```

get the first line of the file

this must be the PDB file with the fewest lines, since sorted-lengths holds files and line counts in order from least to greatest

not particularly obvious

```
$ sort lengths | head -1
  9  methane.pdb
$
```

```
$ sort lengths | head -1
  9  methane.pdb
$
```

a *pipe*

```
$ sort lengths | head -1
  9  methane.pdb
$
```

a *pipe*

use output of left side

```
$ sort lengths | head -1
  9  methane.pdb
$
```

a *pipe*

use output of left side

as input to right side

```
$ sort lengths | head -1
  9  methane.pdb
$
```

a *pipe*

use output of left side

as input to right side

without creating temporary file

```
$ wc -l *.pdb | sort | head -1
  9  methane.pdb
$
```

don't need to create lengths file

```
$ wc -l *.pdb | sort | head -1
  9  methane.pdb
$
```

This simple idea is why Unix has been so successful

```
$ wc -l *.pdb | sort | head -1
  9  methane.pdb
$
```

This simple idea is why Unix has been so successful

Create simple tools that:

```
$ wc -l *.pdb | sort | head -1
  9  methane.pdb
$
```

This simple idea is why Unix has been so successful

Create simple tools that:

– do one job well

```
$ wc -l *.pdb | sort | head -1
  9  methane.pdb
$
```

This simple idea is why Unix has been so successful

Create simple tools that:

– do one job well

– work well with each other

```
$ wc -l *.pdb | sort | head -1
  9  methane.pdb
$
```

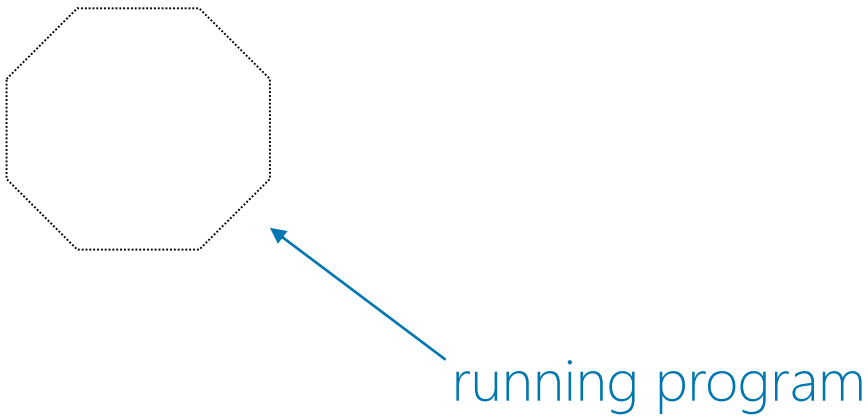
This simple idea is why Unix has been so successful

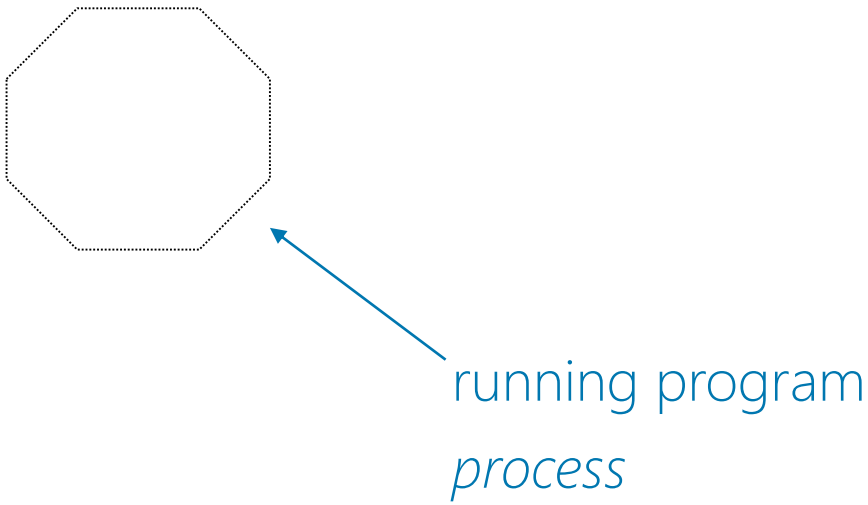
Create simple tools that:

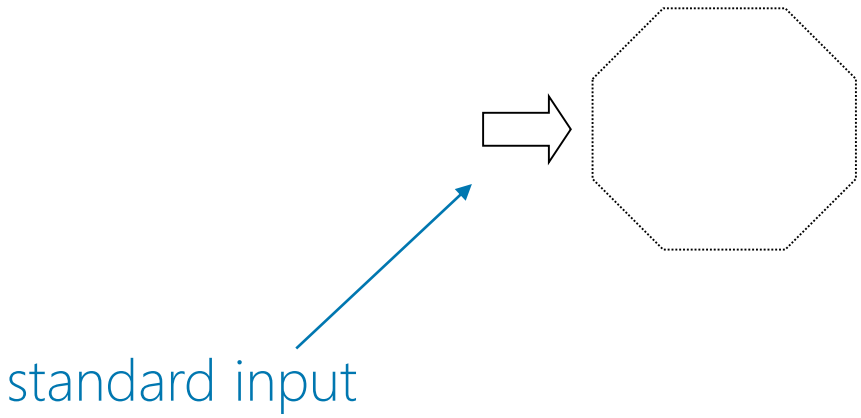
– do one job well

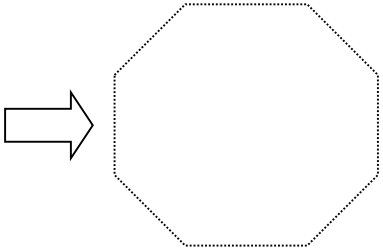
– work well with each other

10 tools can be combined in 100 ways


running program

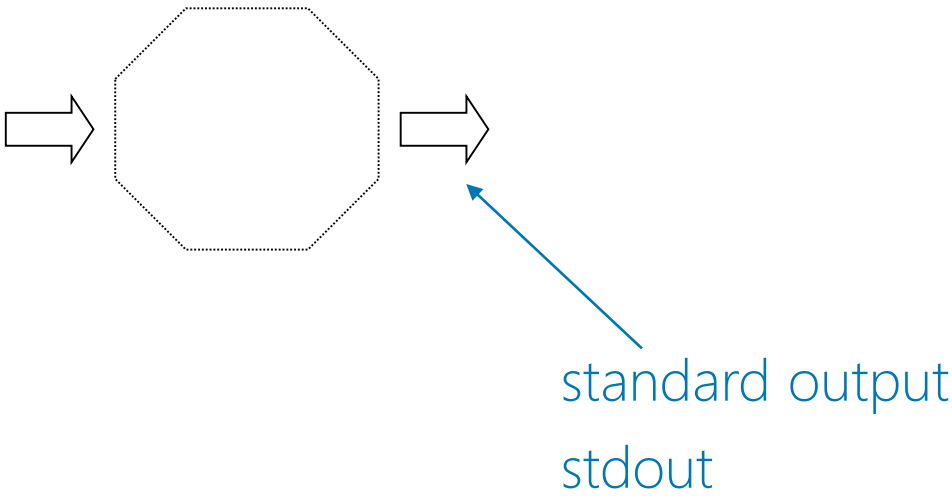

running program
process

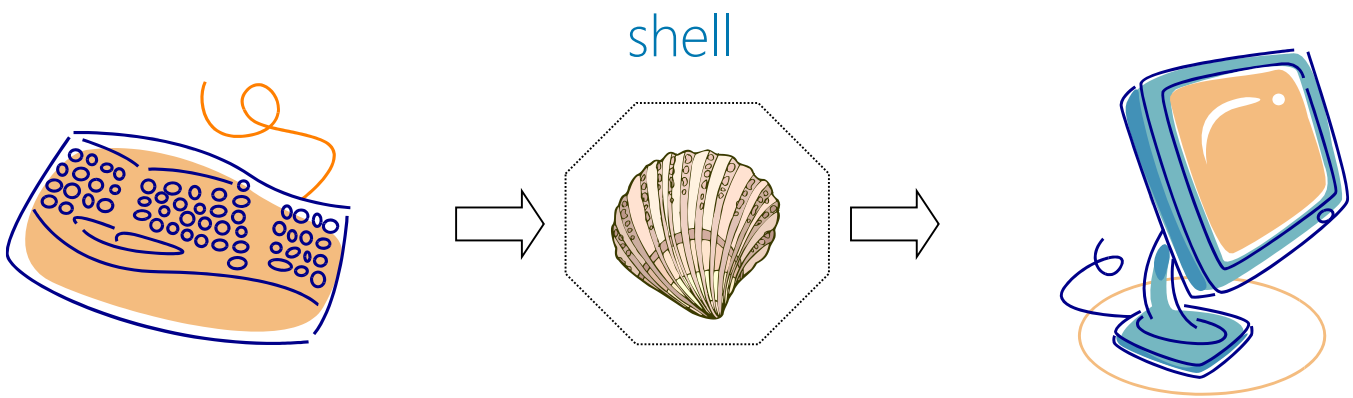

standard input

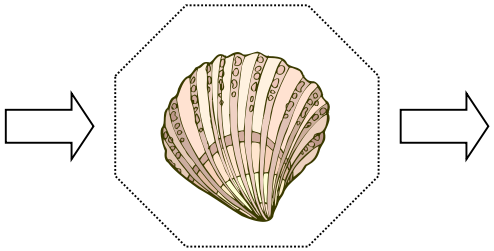


standard input

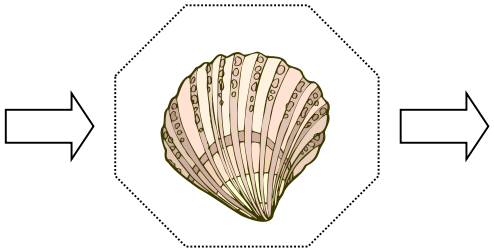
stdin


standard output
stdout

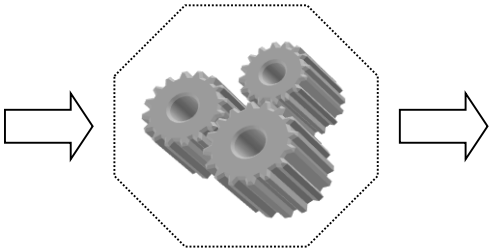

shell



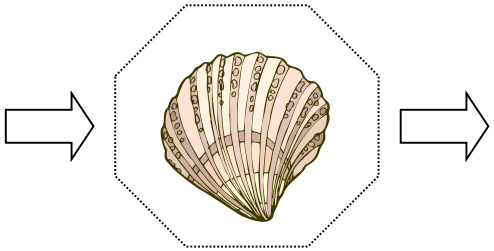
```
$ wc -l *.pdb > lengths
```



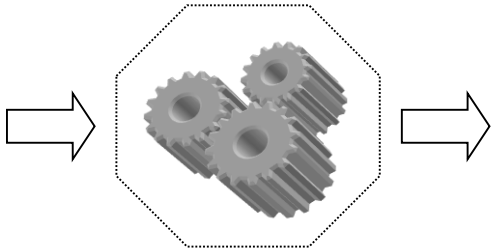
wc



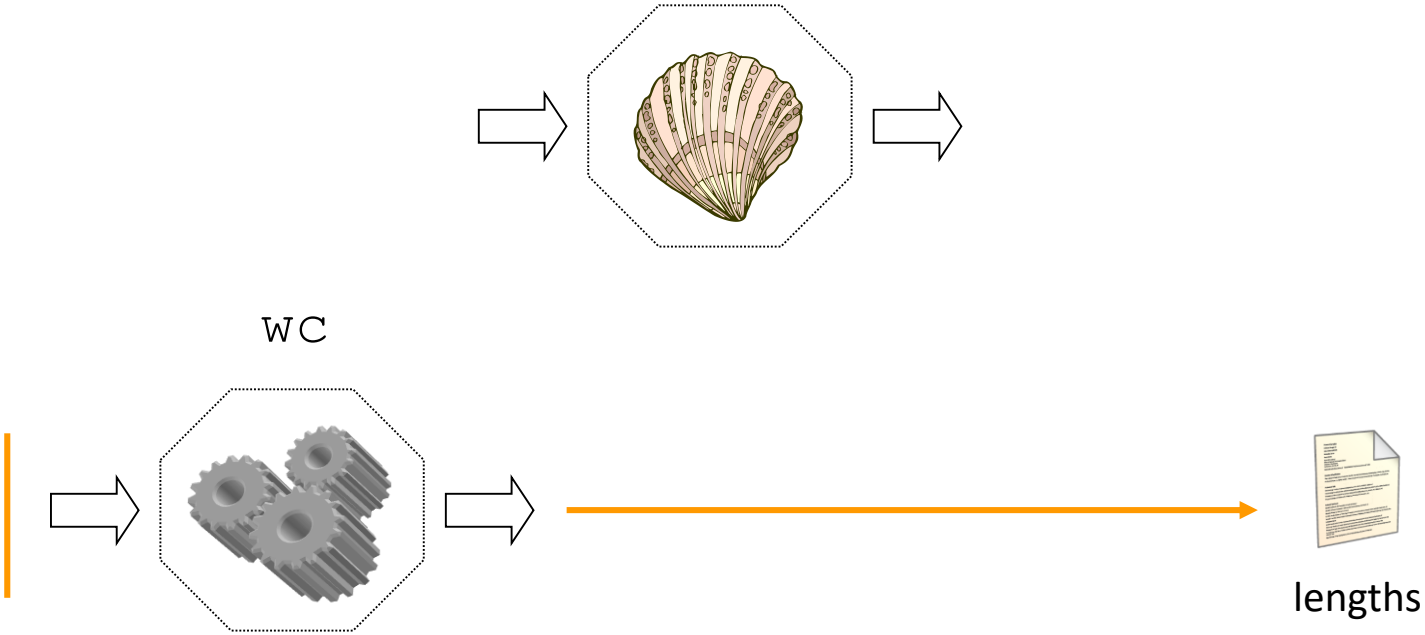
```
$ wc -l *.pdb > lengths
```



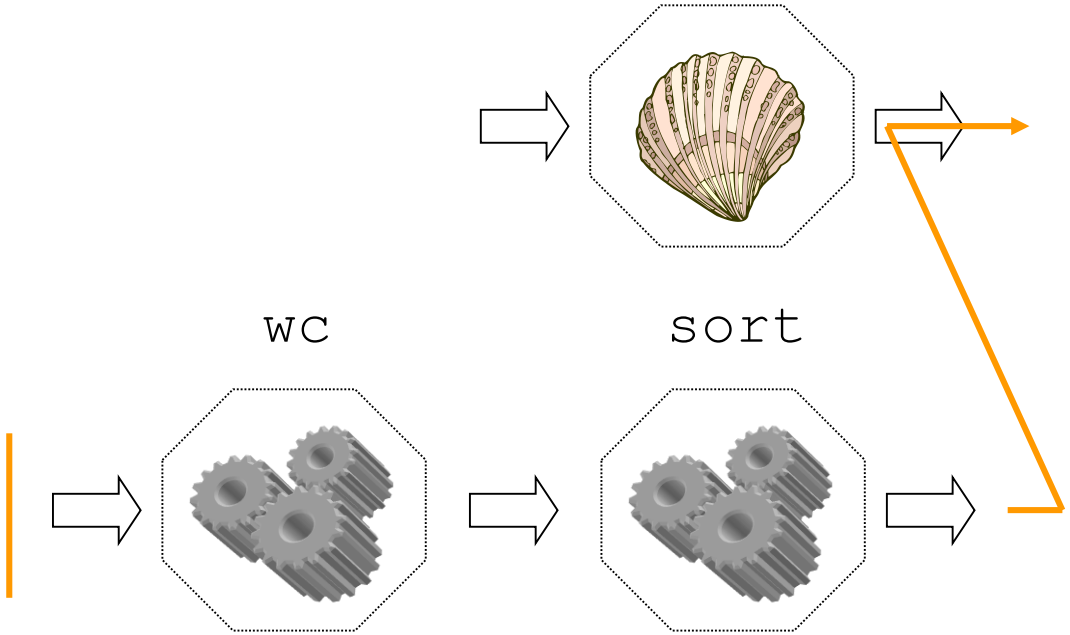
wc



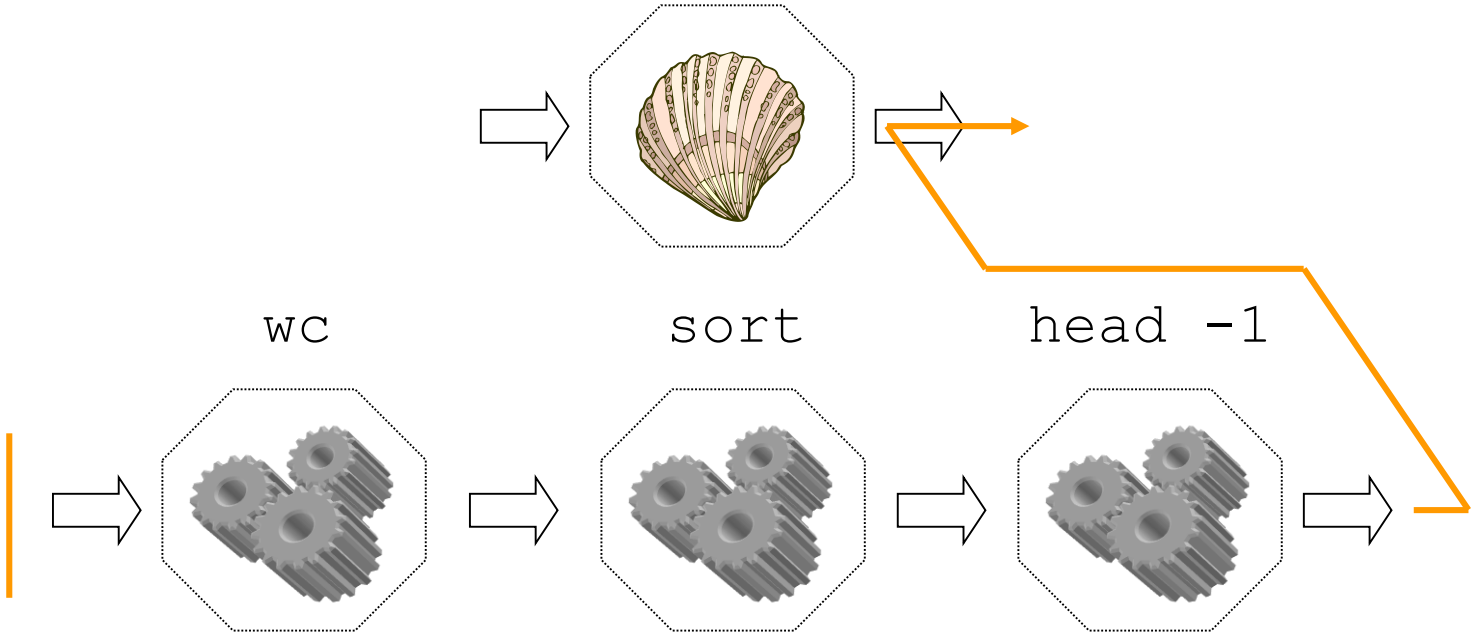
```
$ wc -l *.pdb > lengths
```




```
$ wc -l *.pdb > lengths
```




```
$ wc -l *.pdb | sort
```




```
$ wc -l *.pdb | sort | head -1
```

This programming model called *pipes and filters*

This programming model called *pipes and filters*

A *filter* transforms a stream of input into a stream of output

This programming model called *pipes and filters*

A *filter* transforms a stream of input into a stream of output

A *pipe* connects two filters

This programming model called *pipes and filters*

A *filter* transforms a stream of input into a stream of output

A *pipe* connects two filters

Any program that reads lines of text from standard input, and writes lines of text to standard output, can work with every other

This programming model called *pipes and filters*

A *filter* transforms a stream of input into a stream of output

A *pipe* connects two filters

Any program that reads lines of text from standard input, and writes lines of text to standard output, can work with every other

You can (and should) write such programs

`pwd`	`mkdir`
`cd`	`nano`
`ls`	`rm`
`.`	`rmdir`
`..`	`mv`
	`cp`

pwd	mkdir	wc
cd	nano	sort
ls	rm	head
.	rmdir	
..	mv	
	cp	

pwd	mkdir	wc
cd	nano	sort
ls	rm	head
.	rmdir	*tail*
..	mv	*split*
	cp	*cut*
		uniq

pwd	mkdir	wc	*
cd	nano	sort	>
ls	rm	head	|
.	rmdir	*tail*	
..	mv	*split*	
	cp	*cut*	
		uniq	

pwd	mkdir	wc	*
cd	nano	sort	>
ls	rm	head	|
.	rmdir	*tail*	<
..	mv	*split*	?
	cp	*cut*	
		uniq	