Introduction to:
The Unix Command Line

Including Linux, Mac, other Posix'ly

•••

No experience needed

Why the Command Line…

Access, Power, Control.

But really it’s easier . . .

Overview

- The File Tree and Moving Around
    - Looking at Files and Directories
- Creating and Editing Files
- Getting help (the instructions)
- Stdin, Stdout, Stderr
- Pipes
- History
- Searching
- Remoting
    - Logining in

Sources

- The Linux Documentation Project http://tldp.org/
  - http://tldp.org/guides.html
- Everything else
  - http://tldp.org/LDP/intro-linux/html/index.html
  - http://refspecs.linuxfoundation.org/fhs.shtml
  - http://moo.nac.uci.edu/~hjm/biolinux/Linux_Tutorial_12.html#_moving_bigdata
  - http://moo.nac.uci.edu/~hjm/biolinux/intro-to-linux+hpc-02-16-2018.pdf
  - http://pubs.opengroup.org/onlinepubs/9699919799.2016edition/basedefs/contents.html
  - https://brandonwamboldt.ca/how-linux-pipes-work-under-the-hood-1518/

What is The Command Line?

A Text Based World

- Inputs & Output are strings of characters.
- Everything is a file (or file like)
- The Input from user is the keyboard.
- Operations are initiated sequentially
- Easier to create code for.
- Has a ‘Command Prompt’
    - Usually ‘>’ or ‘$’

Getting started.

Logging on with the Terminal

Linux or Mac open a terminal.
(Git-bash or a Anaconda on windows)

At the prompt ‘$’ or ‘>’ type:

Windows with PuTTY, In
`Host Name (or IP address)`

<*name*>@<*ip-here*>

```
ssh <name>@<ip-here>
```

The Enter the password that goes with your username

“Structure-Function”

The Hierarchical Filesystem Standard

All Unix family systems use a tree based organization.

The top level is called the
“Root Directory”
Represented by a forward slash

/

“Structure-Function”

The Hierarchical Filesystem Standard

All Unix family systems use a tree based organization.

The top level is called the
“Root Directory”
Represented by a forward slash

/

The section for individual accounts
is the “home directory”

/home

With individuals as:

/home/<*loginname*>

Where Are We?

On Windows

C:\Users\<username>

Might look like

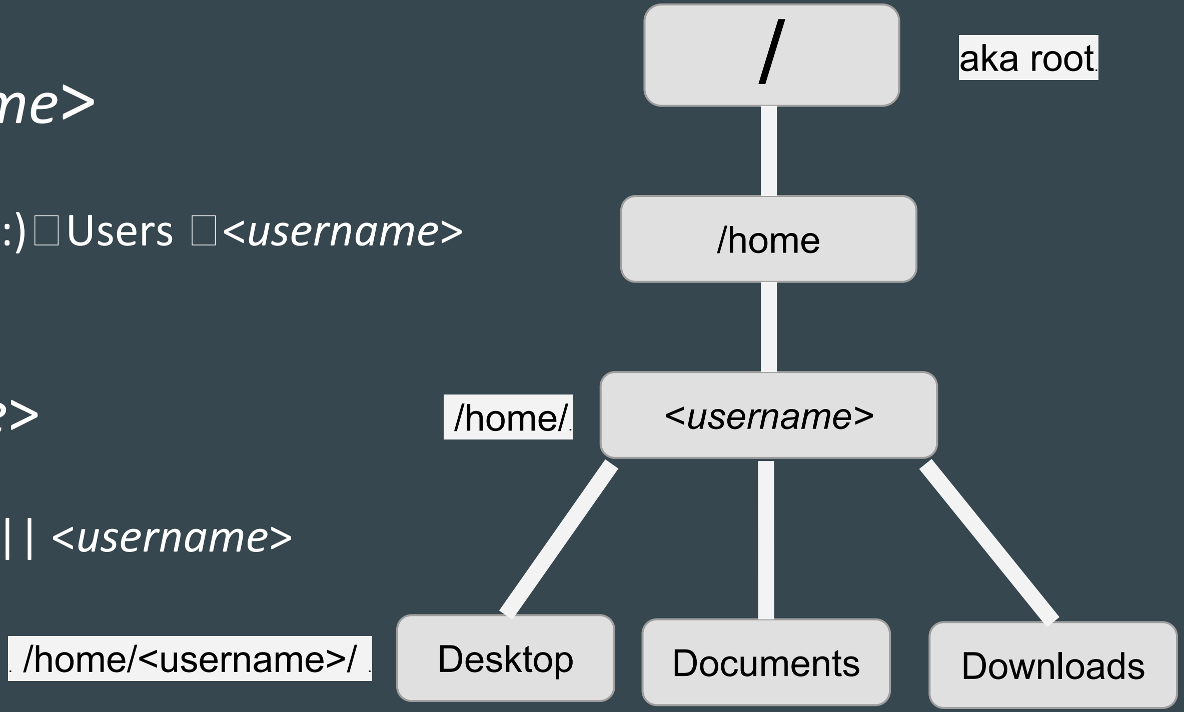
Computer□Local Disk (C:)□Users □*<username>*

On Mac

/Users/*<username>*

Might look like

Macintosh HD || Users || *<username>*



/

aka root

/home

/home/

<username>

/home/<username>/

Desktop

Documents

Downloads

On Unix, Linux etc.

`/home/<username>`

Where Are We?

On Windows

C:\Users\\<*username*>

Might look like

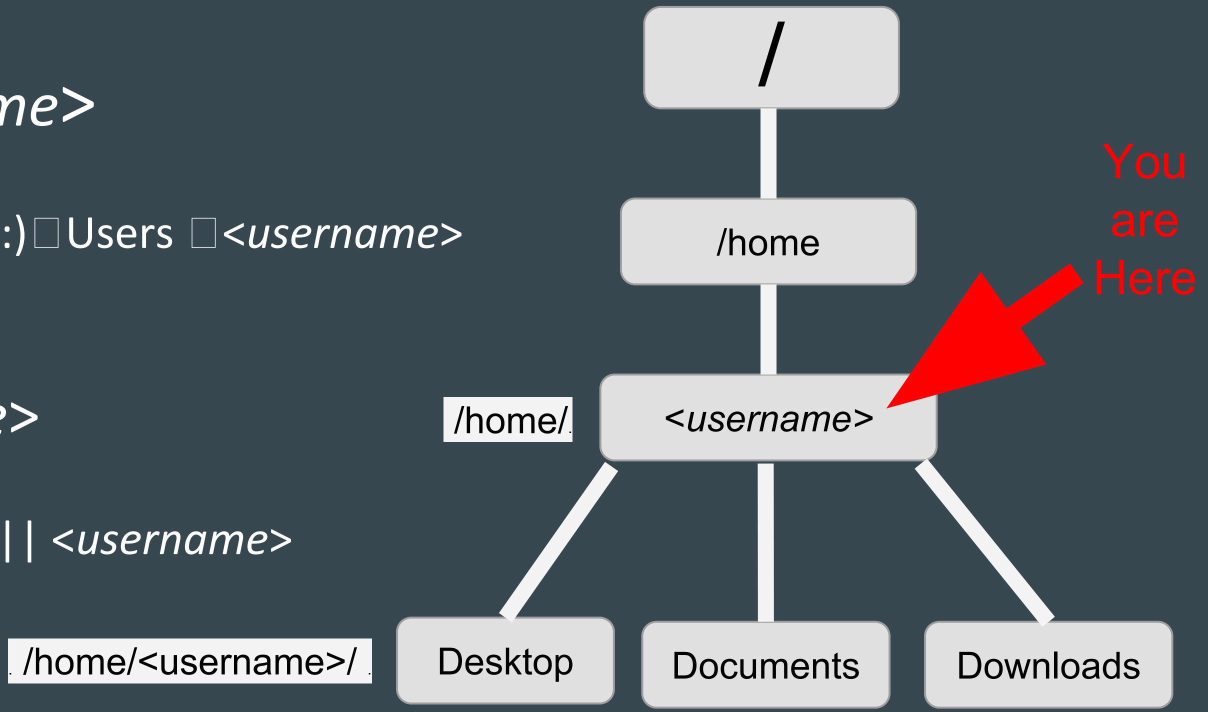
Computer□Local Disk (C:)□Users □<*username*>

On Mac

/Users/<*username*>

Might look like

Macintosh HD || Users || <*username*>



On Unix, Linux etc.

/home/<*username*>

Let’s check

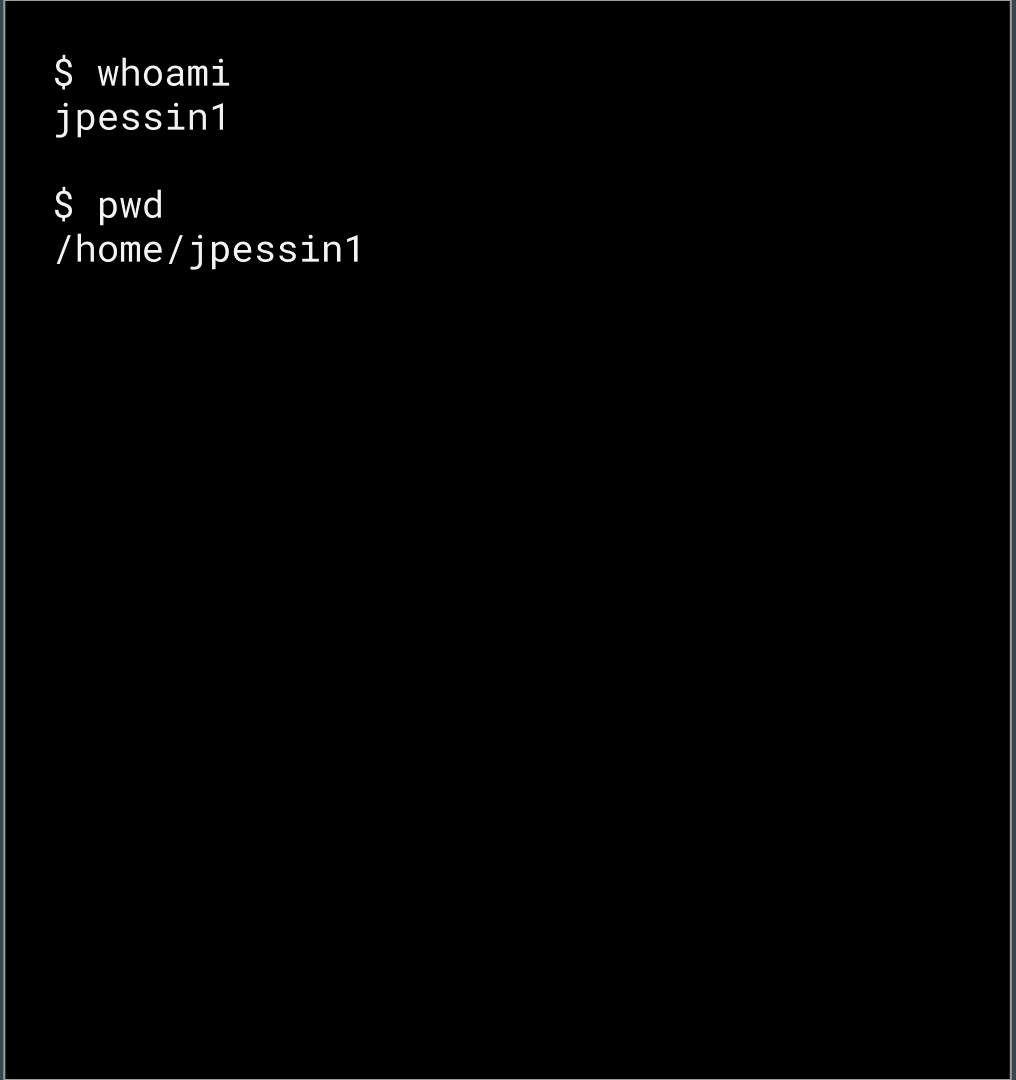
whoami & pwd

`whoami`

prints the login name you are currently using

`pwd`

prints the name of the current directory



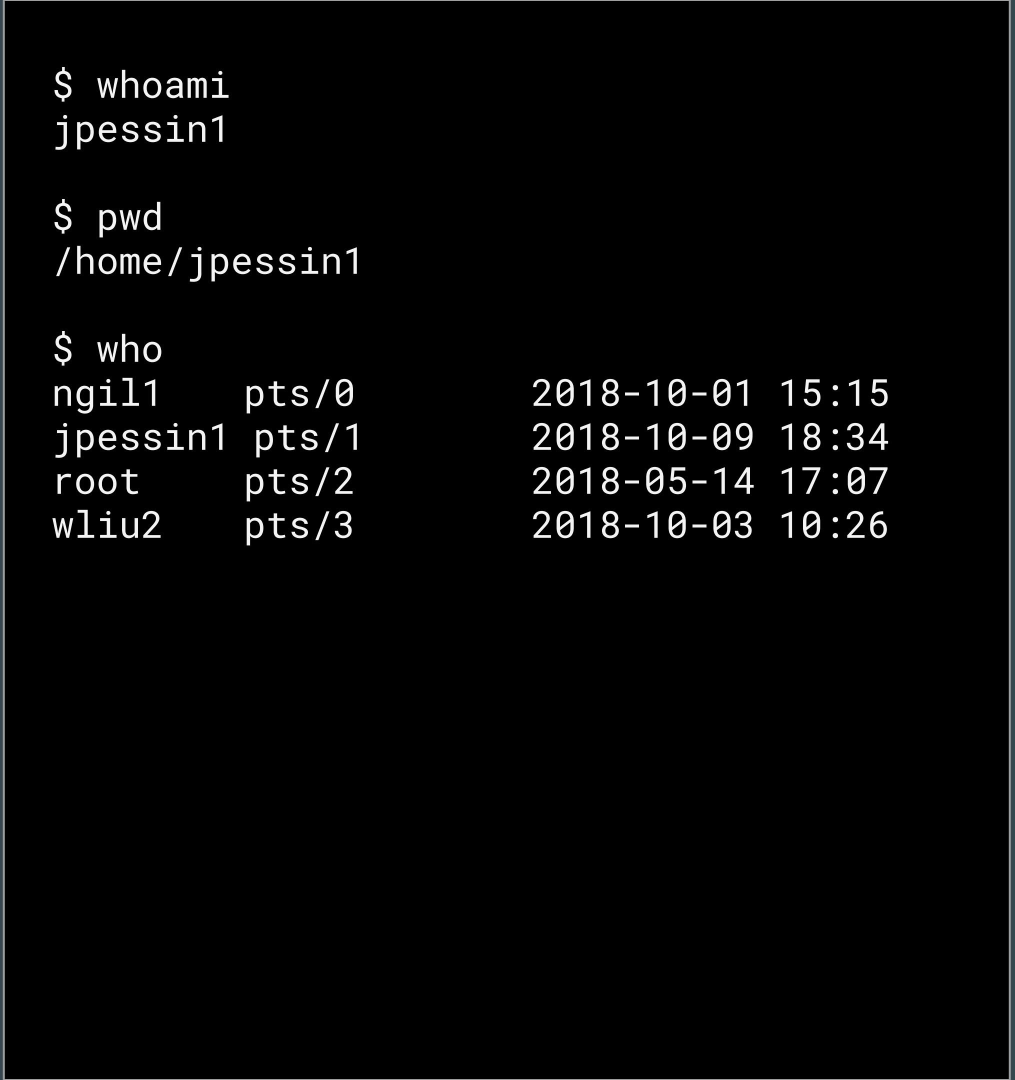
```
$ whoami
jpessin1

$ pwd
/home/jpessin1
```

Who else is here and where are they from?

who & w

who

Prints info about who is logged-in



```
$ whoami
jpessin1

$ pwd
/home/jpessin1

$ who
ngil1    pts/0        2018-10-01 15:15
jpessin1 pts/1        2018-10-09 18:34
root     pts/2        2018-05-14 17:07
wliu2    pts/3        2018-10-03 10:26
```

Who else is here and where are they from?

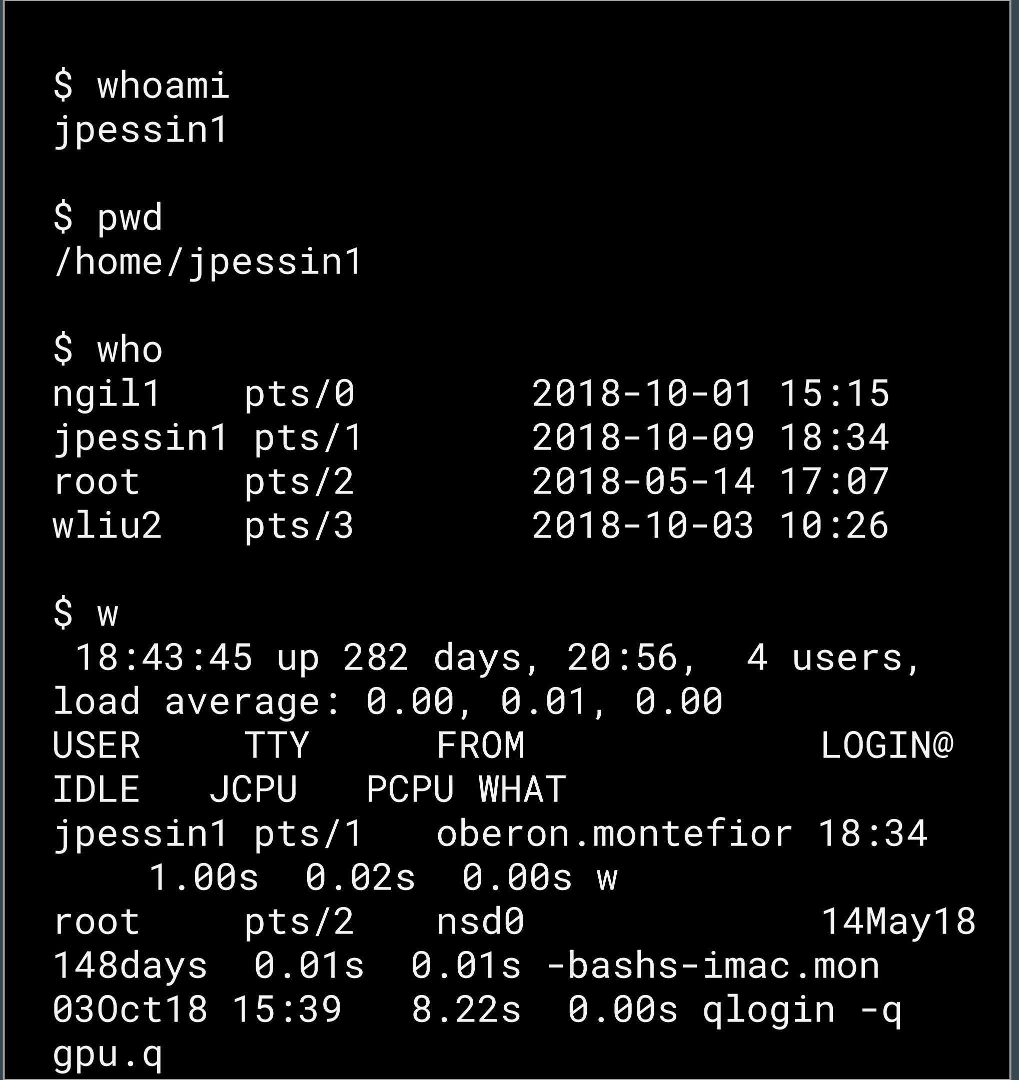
who & w

who

Prints info about who is logged-in

w

More information about current users



```
$ whoami
jpessin1

$ pwd
/home/jpessin1

$ who
ngil1    pts/0        2018-10-01 15:15
jpessin1 pts/1        2018-10-09 18:34
root     pts/2        2018-05-14 17:07
wliu2    pts/3        2018-10-03 10:26

$ w
 18:43:45 up 282 days, 20:56,  4 users,
load average: 0.00, 0.01, 0.00
USER     TTY     FROM             LOGIN@
IDLE   JCPU   PCPU WHAT
jpessin1 pts/1   oberon.montefior 18:34
    1.00s  0.02s  0.00s w
root     pts/2   nsd0             14May18
148days  0.01s  0.01s -bashs-imac.mon
03Oct18 15:39   8.22s  0.00s qlogin -q
gpu.q
```

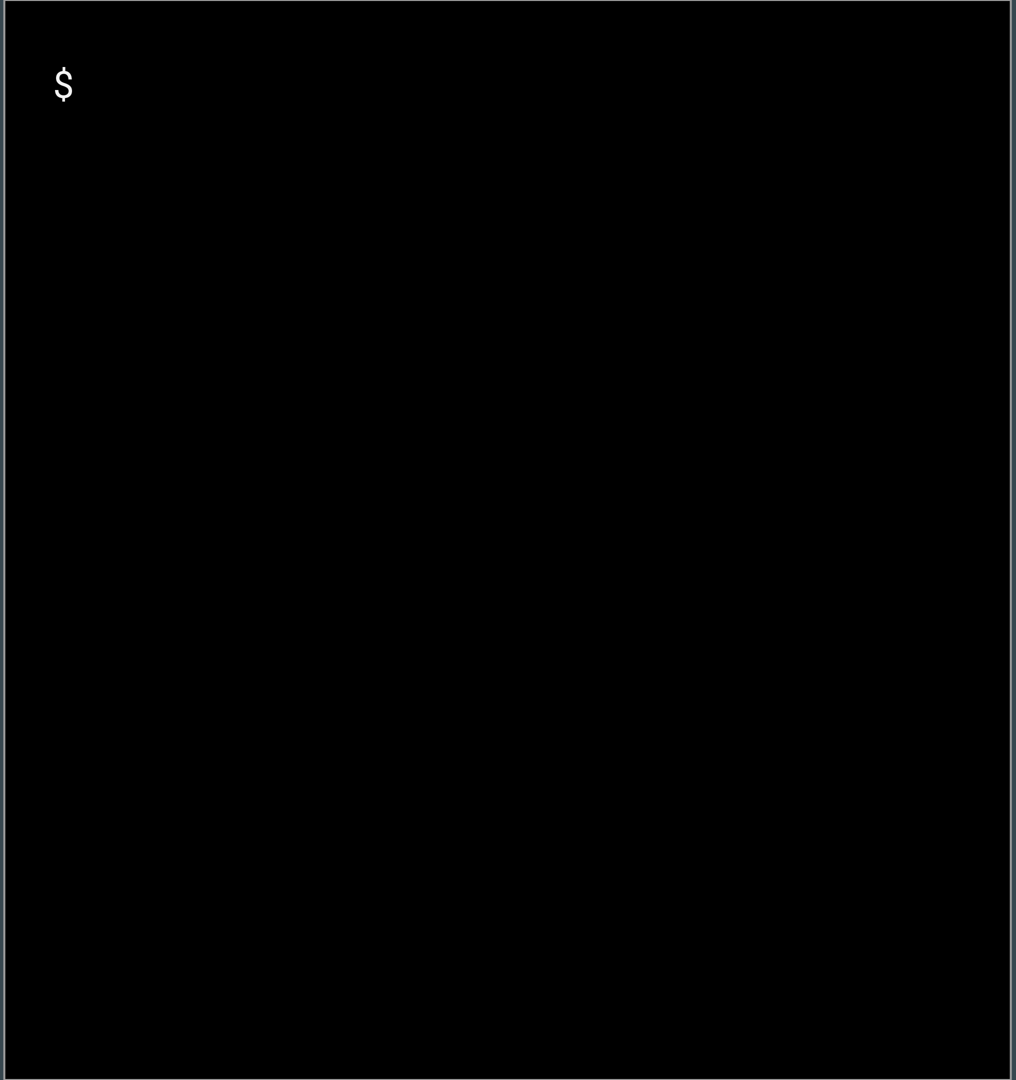
Let's clean up

clear

```
clear
```

Clears the screen,

Well, it prints the exact number of lines to push everything off the screen.



```
$
```

w again, with tokens

w, Options

w -s -h

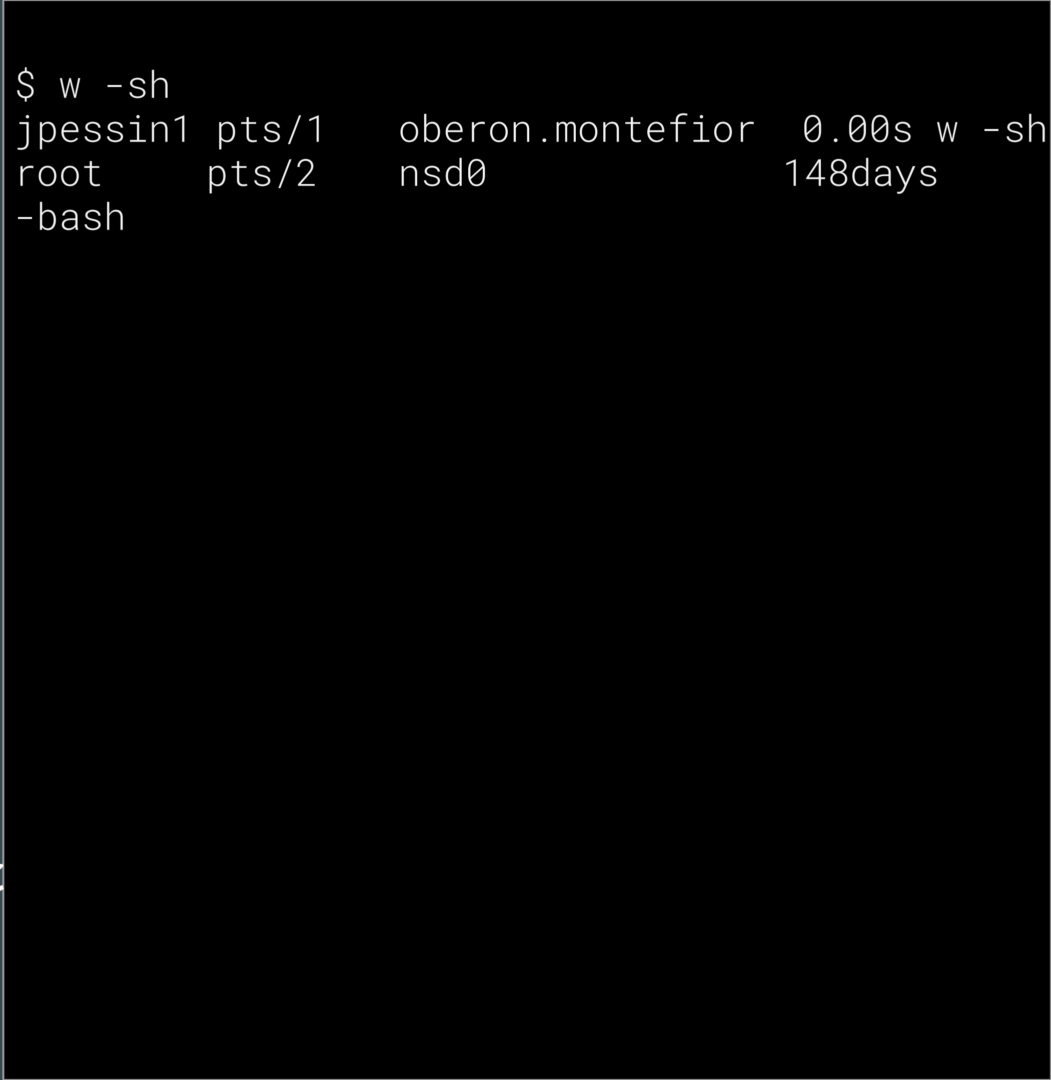
Same command with some options specified.

In this case

-s means: use the short format.

-h means: Don’t print the header.

For many commands you can put the option together



```
$ w -sh
jpessin1 pts/1    oberon.montefior  0.00s w -sh
root     pts/2    nsd0                  148days
-bash
```

w again, with tokens

w, Options

```
w -s -h
```

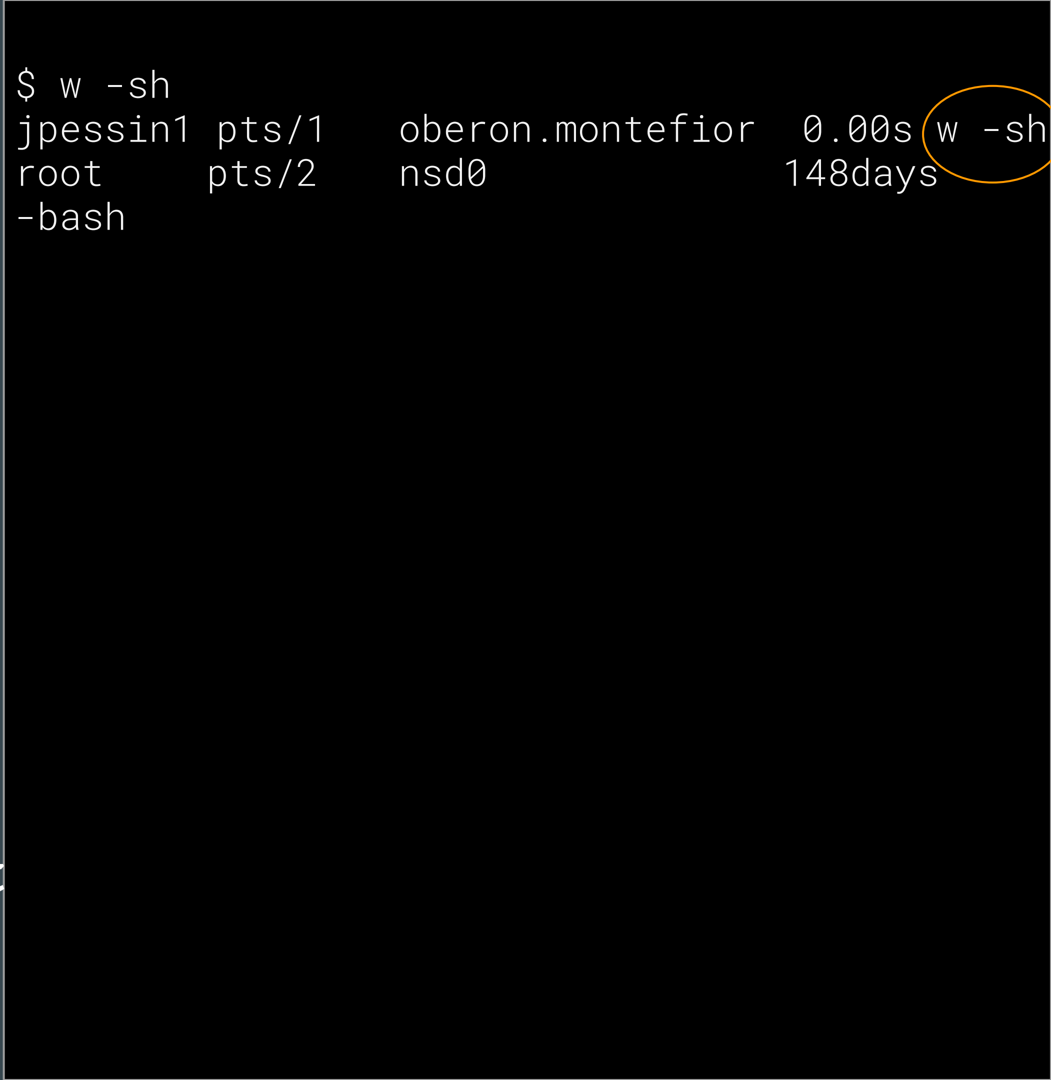
Same command with some options specified.

In this case

-s means: use the short format.

-h means: Don't print the header.

For many commands you can put the option together



```
$ w -sh
jpessin1 pts/1    oberon.montefior   0.00s w -sh
root     pts/2    nsd0              148days
-bash
```

w again, with tokens

w, Options

```
w -s -h
```

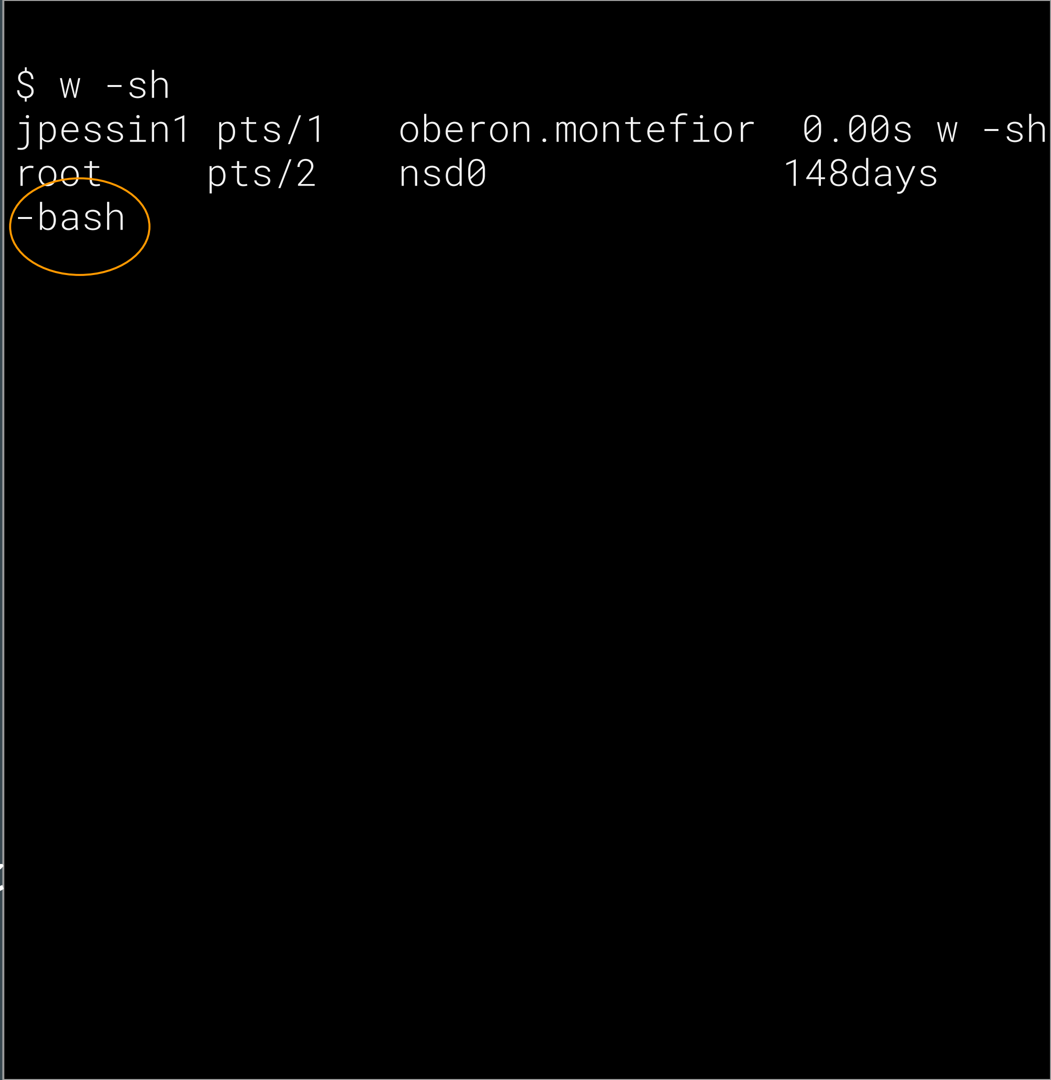
Same command with some options specified.

In this case

-s means: use the short format.

-h means: Don’t print the header.

For many commands you can put the option together



```
$ w -sh
jpessin1 pts/1    oberon.montefior  0.00s w -sh
root     pts/2    nsd0             148days
-bash
```

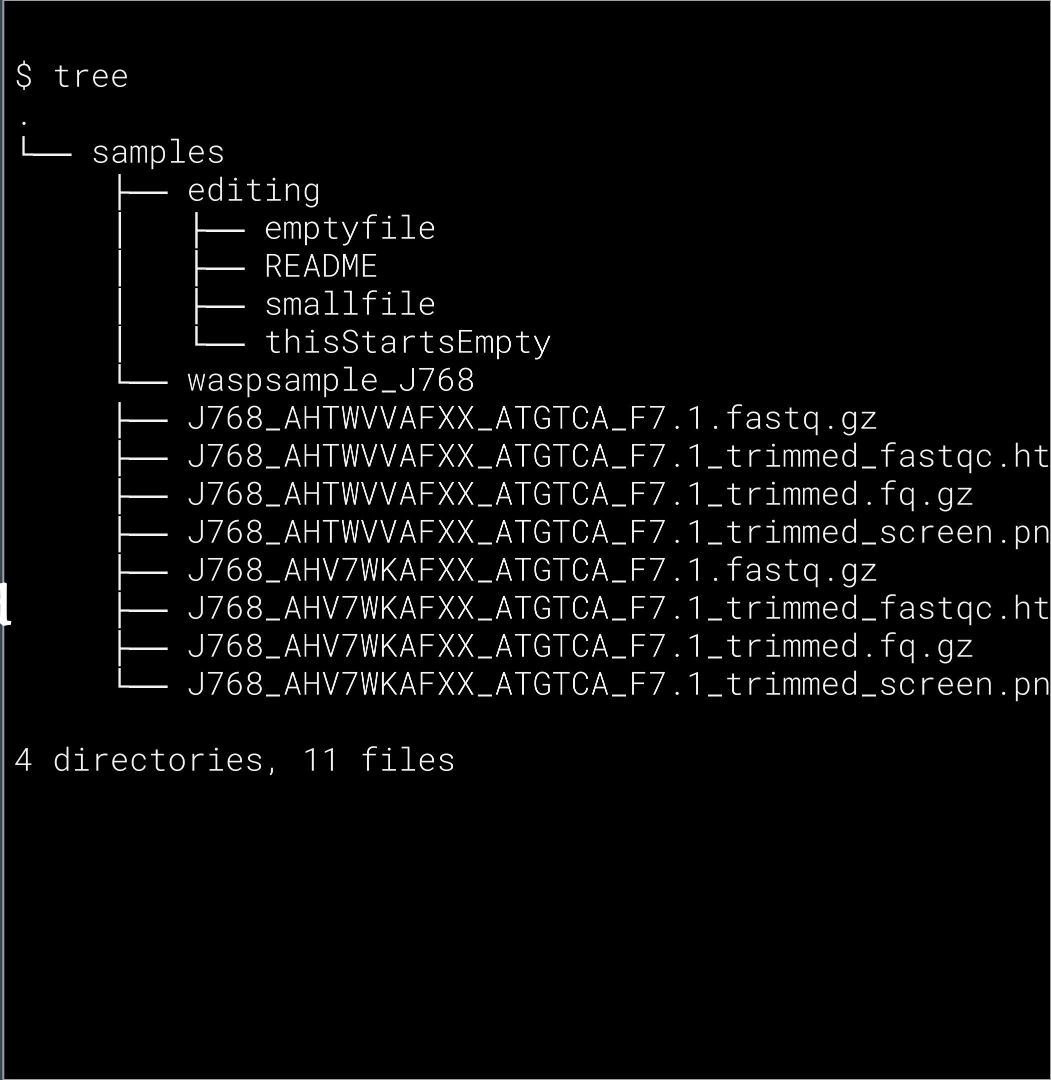
A helpful tree

tree

tree list contents of directories in a tree-like format.

By default it is recursive.

This means it will repeat over and over going down the tree.



```
$ tree
.
└── samples
    ├── editing
    │   ├── emptyfile
    │   ├── README
    │   ├── smallfile
    │   └── thisStartsEmpty
    └── waspsample_J768
    ├── J768_AHTWVVAFXX_ATGTCA_F7.1.fastq.gz
    ├── J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed_fastqc.ht
    ├── J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed.fq.gz
    ├── J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed_screen.pn
    ├── J768_AHV7WKAFXX_ATGTCA_F7.1.fastq.gz
    ├── J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_fastqc.ht
    ├── J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed.fq.gz
    └── J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_screen.pn

4 directories, 11 files
```

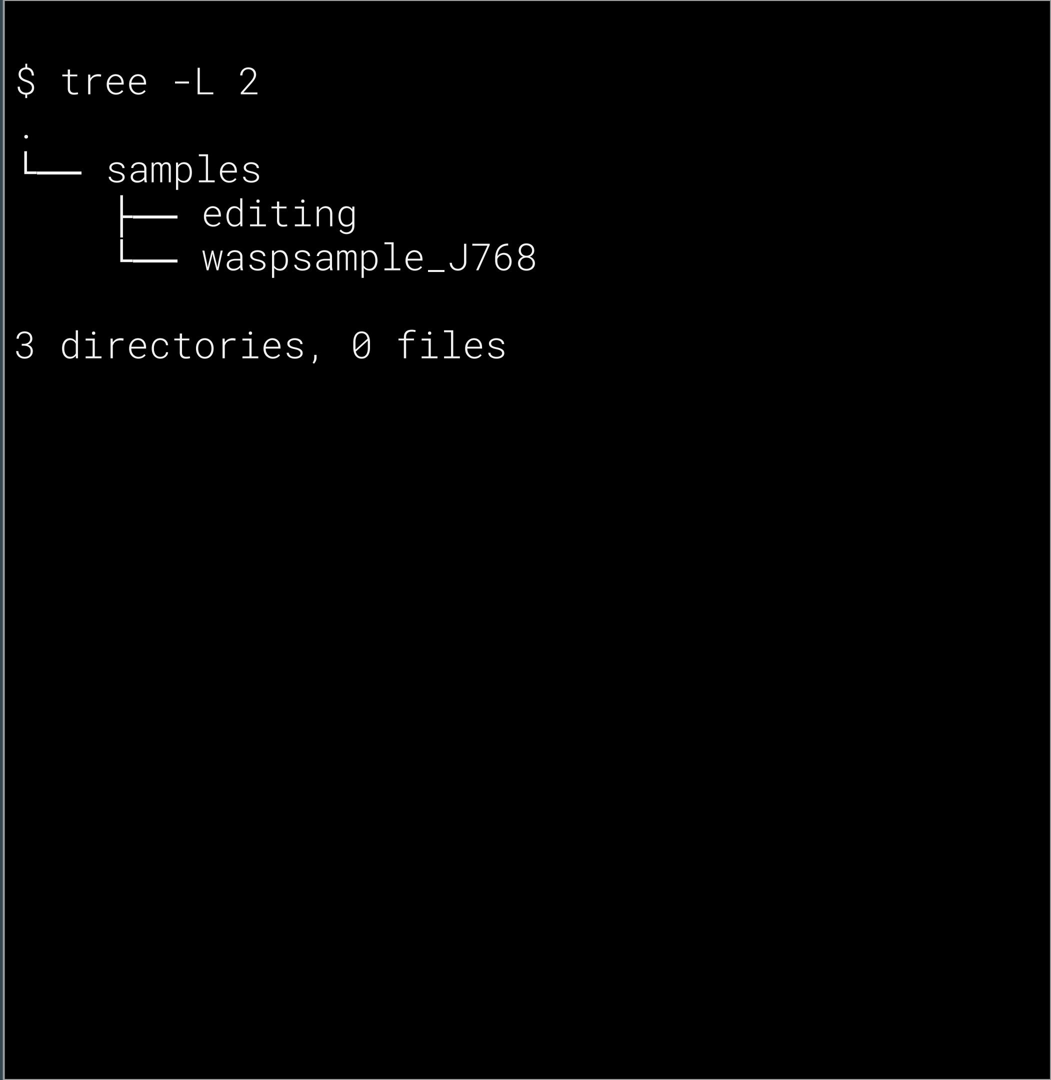
A helpful tree

tree -L number

For tree the -L option also requires a number

```
Tree -L 2
```

Limits the recursion depth how many repetitions to the number.



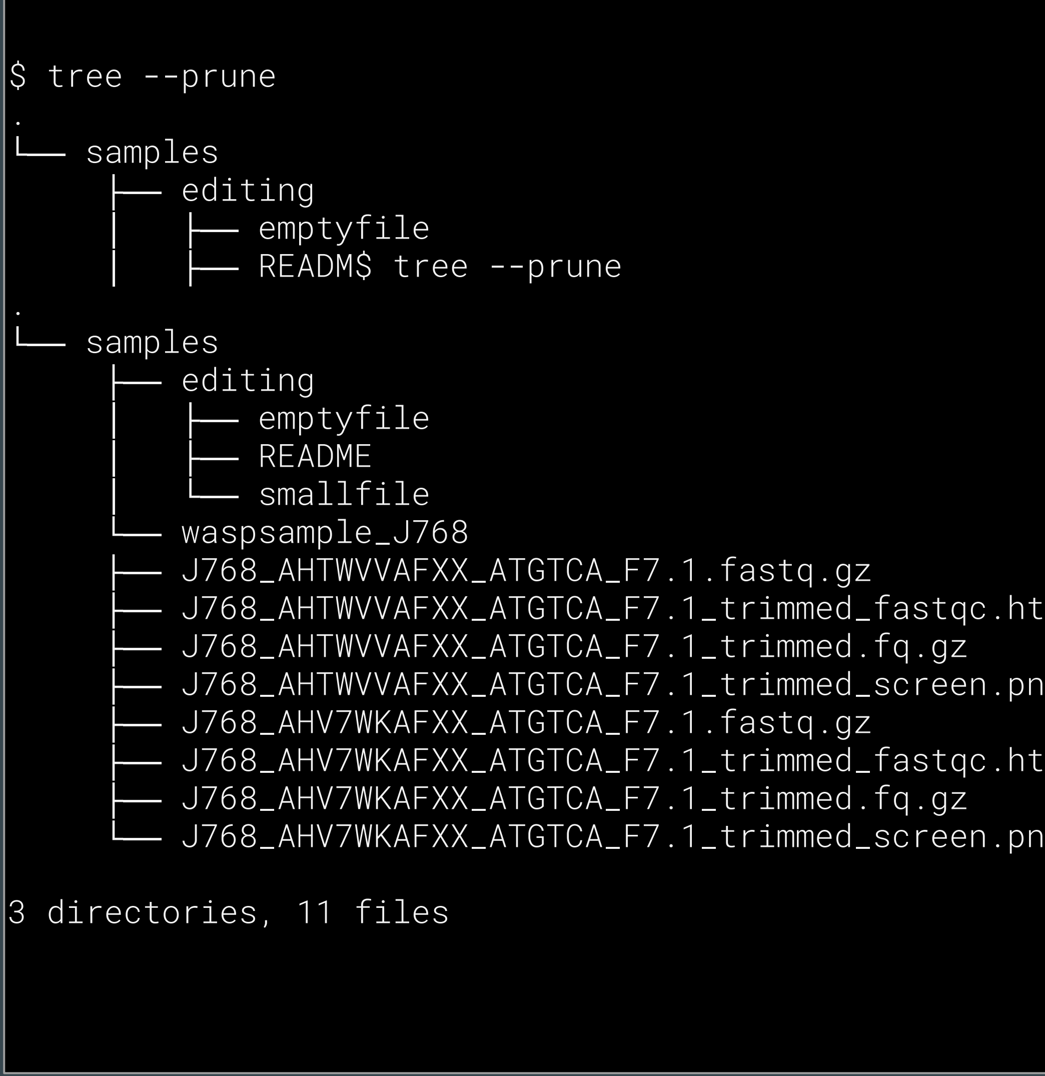
```
$ tree -L 2
.
└── samples
    ├── editing
    └── waspsample_J768

3 directories, 0 files
```

A helpful tree

tree --prune

The --prune option

Makes tree prune empty directories from the output



```
$ tree --prune
.
└── samples
    ├── editing
    │   ├── emptyfile
    │   ├── READM$ tree --prune
.
└── samples
    ├── editing
    │   ├── emptyfile
    │   ├── README
    │   └── smallfile
    └── waspsample_J768
    ├── J768_AHTWVVAFXX_ATGTCA_F7.1.fastq.gz
    ├── J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed_fastqc.ht
    ├── J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed.fq.gz
    ├── J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed_screen.pn
    ├── J768_AHV7WKAFXX_ATGTCA_F7.1.fastq.gz
    ├── J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_fastqc.ht
    ├── J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed.fq.gz
    └── J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_screen.pn

3 directories, 11 files
```

In the Unix universe everything is a file.

Print working directory, list directory content

pwd ls

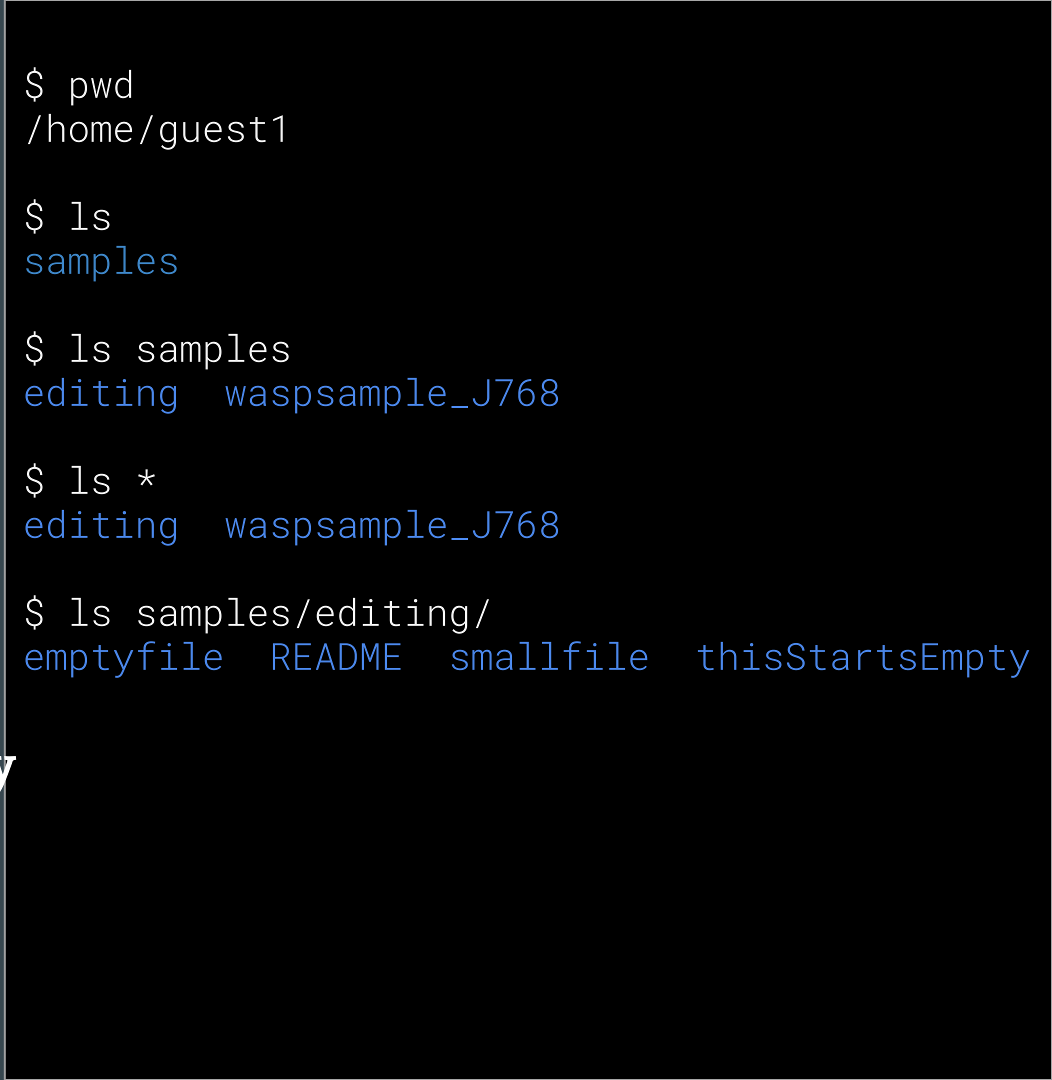
pwd

Prints the working directoy

ls

Lists directory contents

*** Wild card, it substitutes for any combination of characters. (Globbing)**



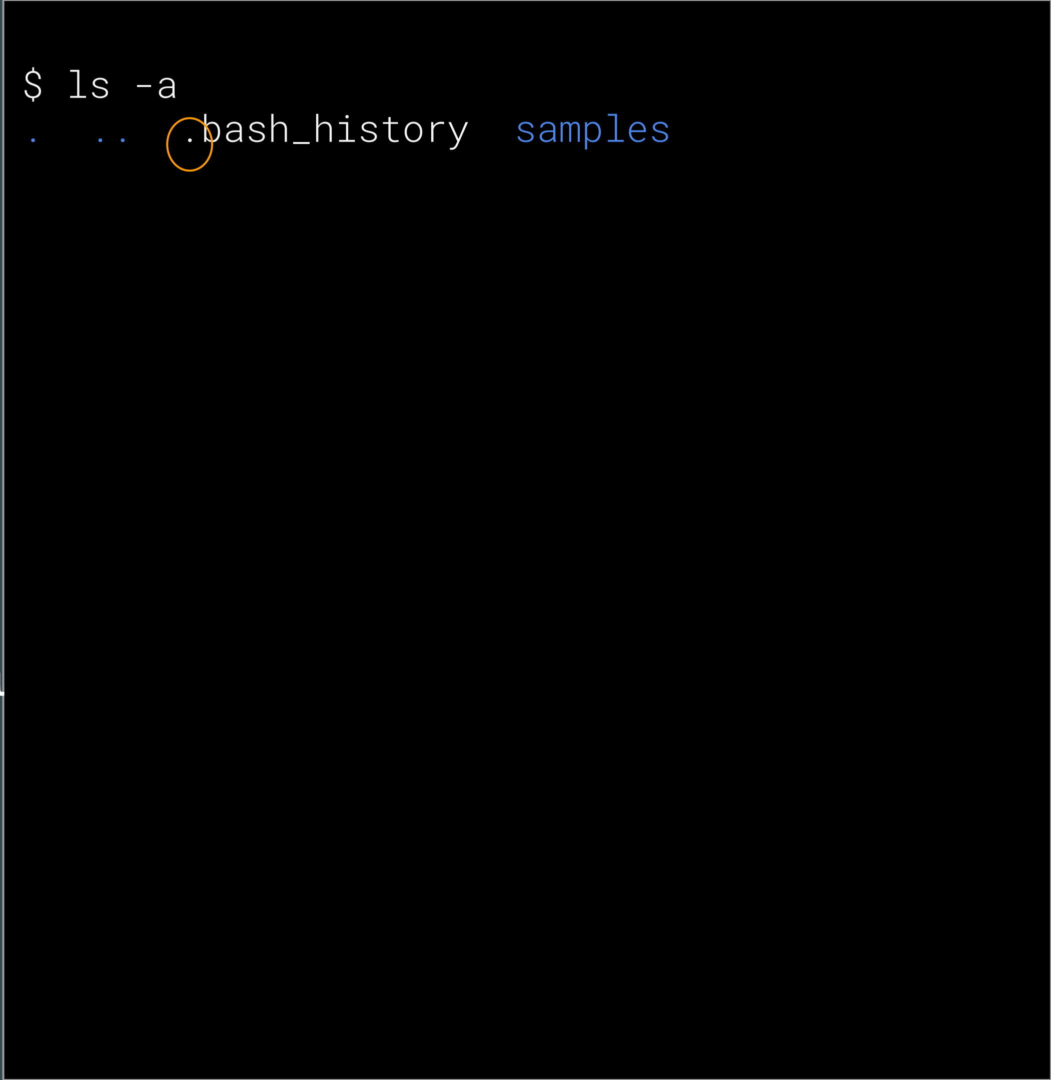
```
$ pwd
/home/guest1

$ ls
samples

$ ls samples
editing  waspsample_J768

$ ls *
editing  waspsample_J768

$ ls samples/editing/
emptyfile  README  smallfile  thisStartsEmpty
```

Common options for ls

ls -a -A -l -h -r -s -t

```
ls
```

-a	show all
-A	show hidden
-l	long form
-h	human readable: w/long form
-r	reverse the order
-s	sort by size
-t	sort by last modified time
-1	one file per line



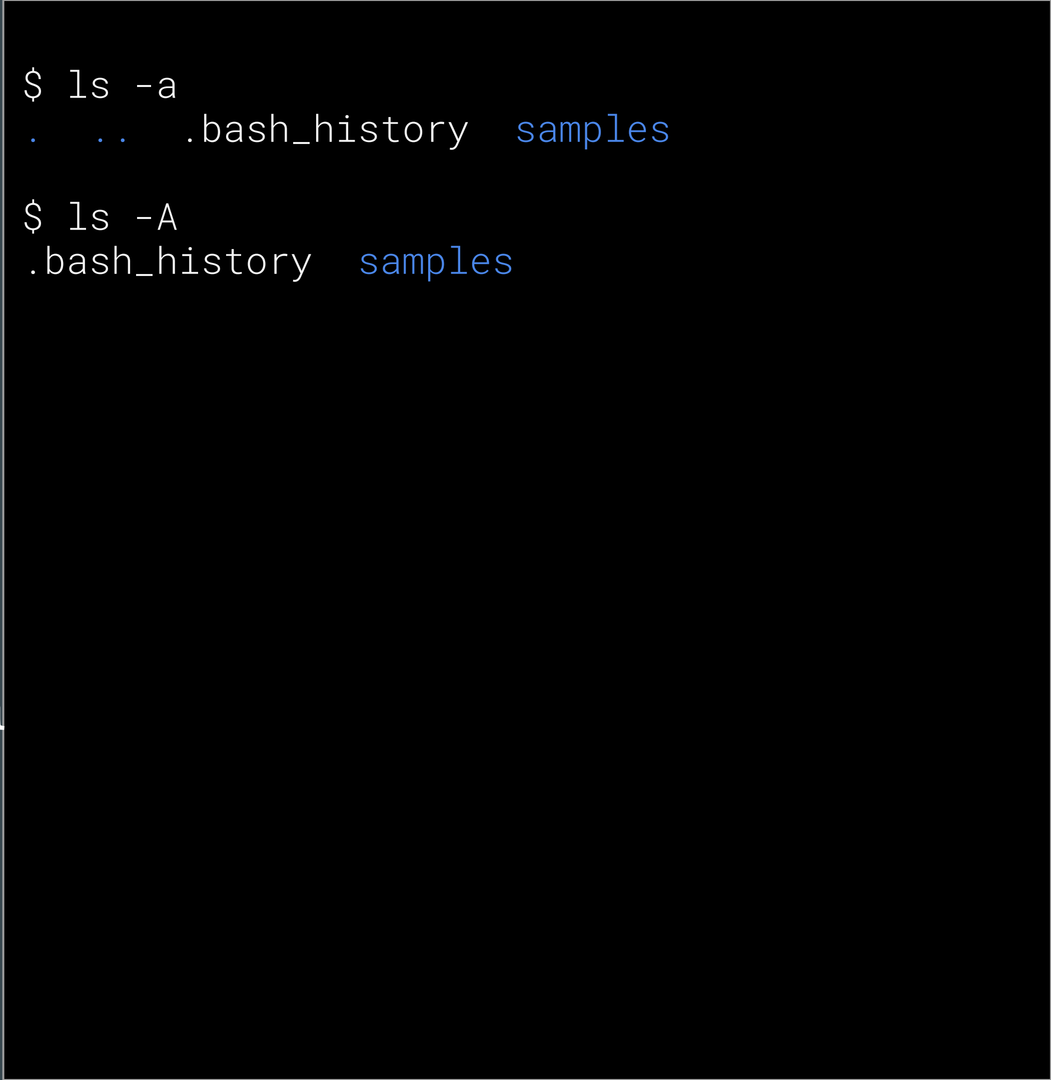
```
$ ls -a
.   ..   .bash_history   samples
```

Common options for ls

ls -a -A -l -h -r -s -t

```
ls
```

-a	show all
-A	show hidden
-F	append indicator
-l	long form
-h	human readable: w/long form
-r	reverse the order
-s	sort by size
-t	sort by last modified time
-1	one file per line



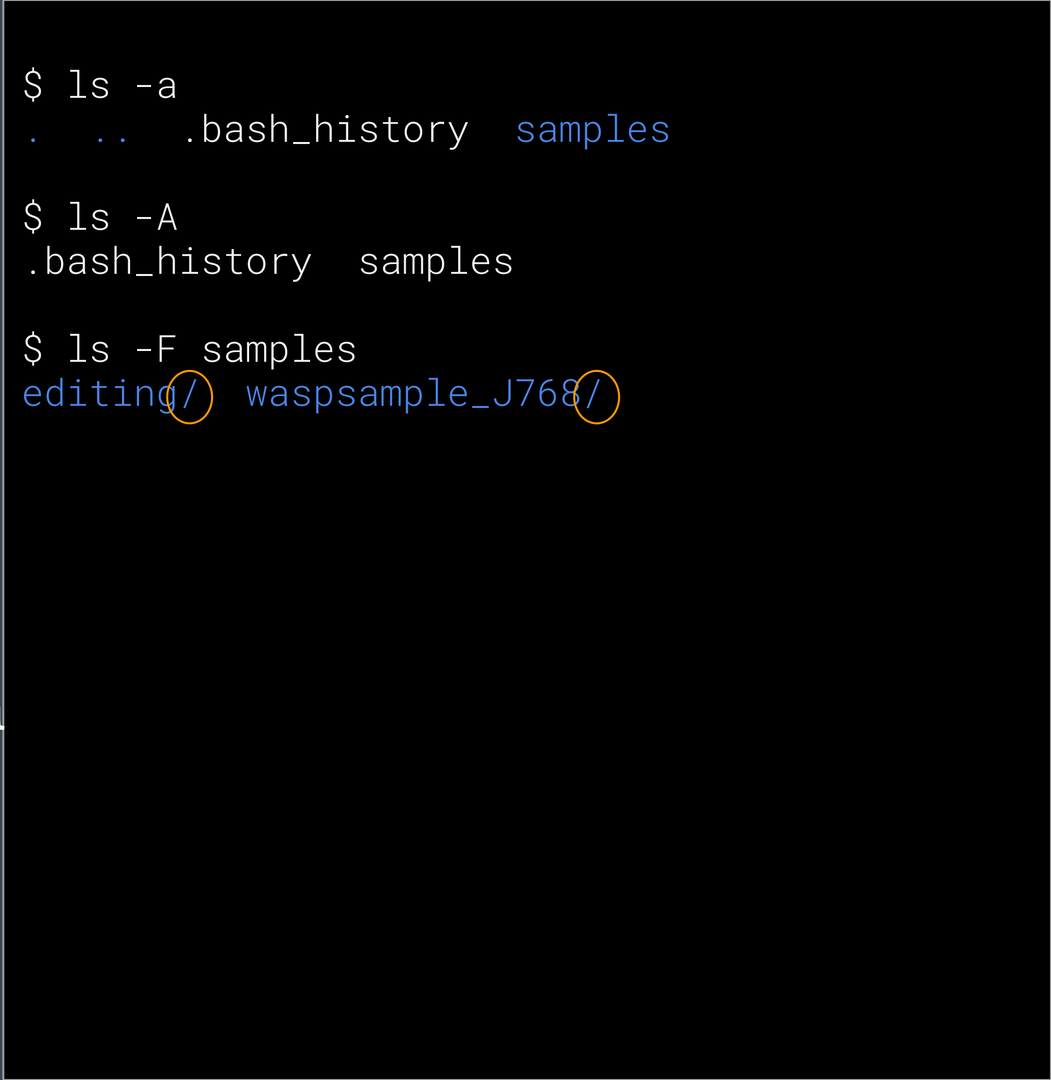
```
$ ls -a
.   ..   .bash_history   samples

$ ls -A
.bash_history   samples
```

Common options for ls

ls -a -A -l -h -r -s -t

ls

-a	show all
-A	show hidden
-F	append indicator
-l	long form
-h	human readable: w/long form
-r	reverse the order
-s	sort by size
-t	sort by last modified time
-1	one file per line



```
$ ls -a
.   ..   .bash_history  samples

$ ls -A
.bash_history  samples

$ ls -F samples
editing/  waspsample_J768/
```

Common options for ls

ls -a -A -l -h -r -s -t

```
ls
```

-a show all

-A show hidden

-F append indicator

-l long form

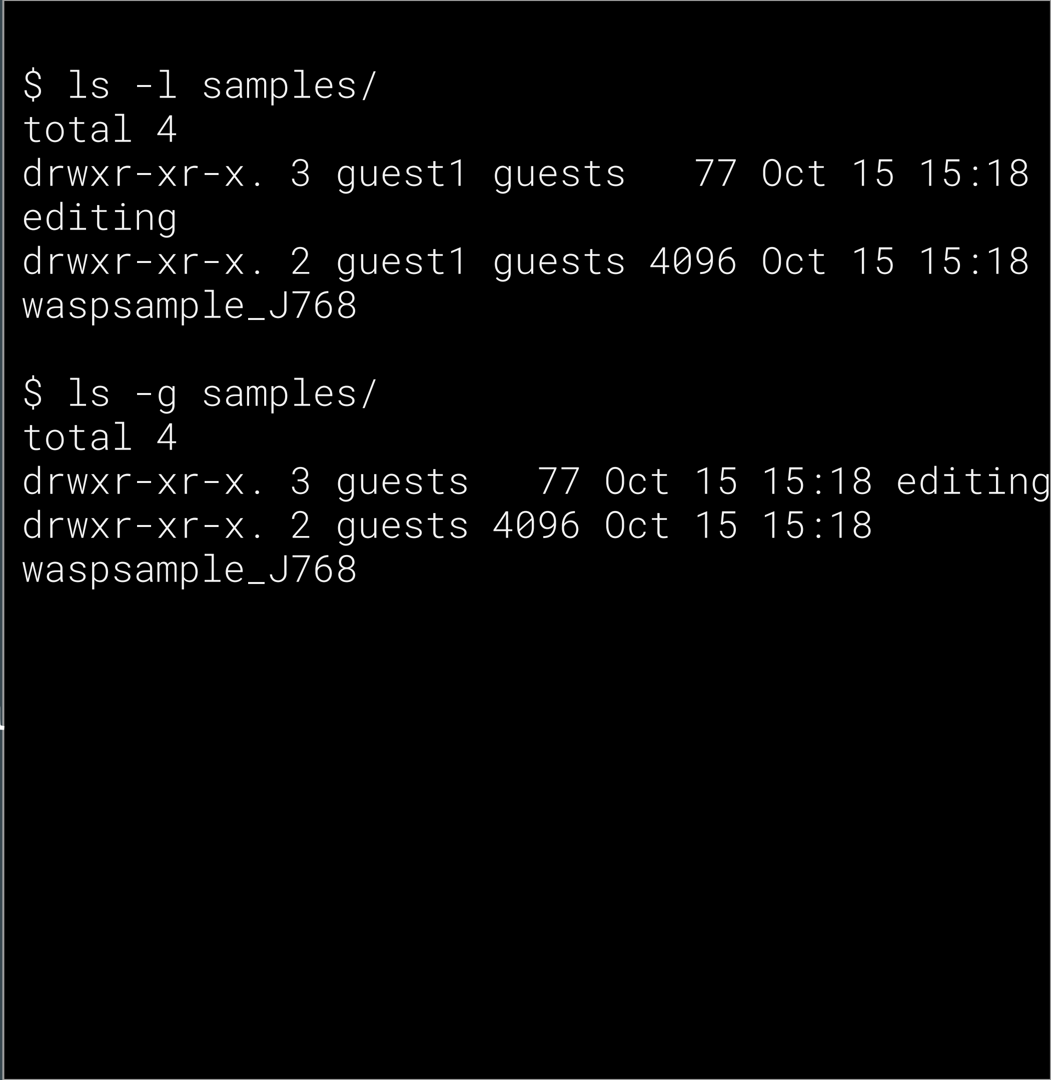
-h human readable: w/long form

-r reverse the order

-s sort by size

-t sort by last modified time

-1 one file per line



```
$ ls -l samples/
total 4
drwxr-xr-x. 3 guest1 guests   77 Oct 15 15:18
editing
drwxr-xr-x. 2 guest1 guests 4096 Oct 15 15:18
waspsample_J768

$ ls -g samples/
total 4
drwxr-xr-x. 3 guests   77 Oct 15 15:18 editing
drwxr-xr-x. 2 guests 4096 Oct 15 15:18
waspsample_J768
```

cd change directory

cd exampledir

`cd <directory>`

Change the current directory.

Directories can be referred to in ‘absolute terms” i. e. Listed explicitly from root like pwd does . . . or relative to your current directory

A blank ‘cd’ will take you back to your home dir

```
$ cd

$ pwd
/home/guest1

$ ls
Samples

$ cd samples

$ ls -F
editing/  waspsample_J768/

$ cd

$ pwd
/home/guest1
```

cd change directory

cd exampledir

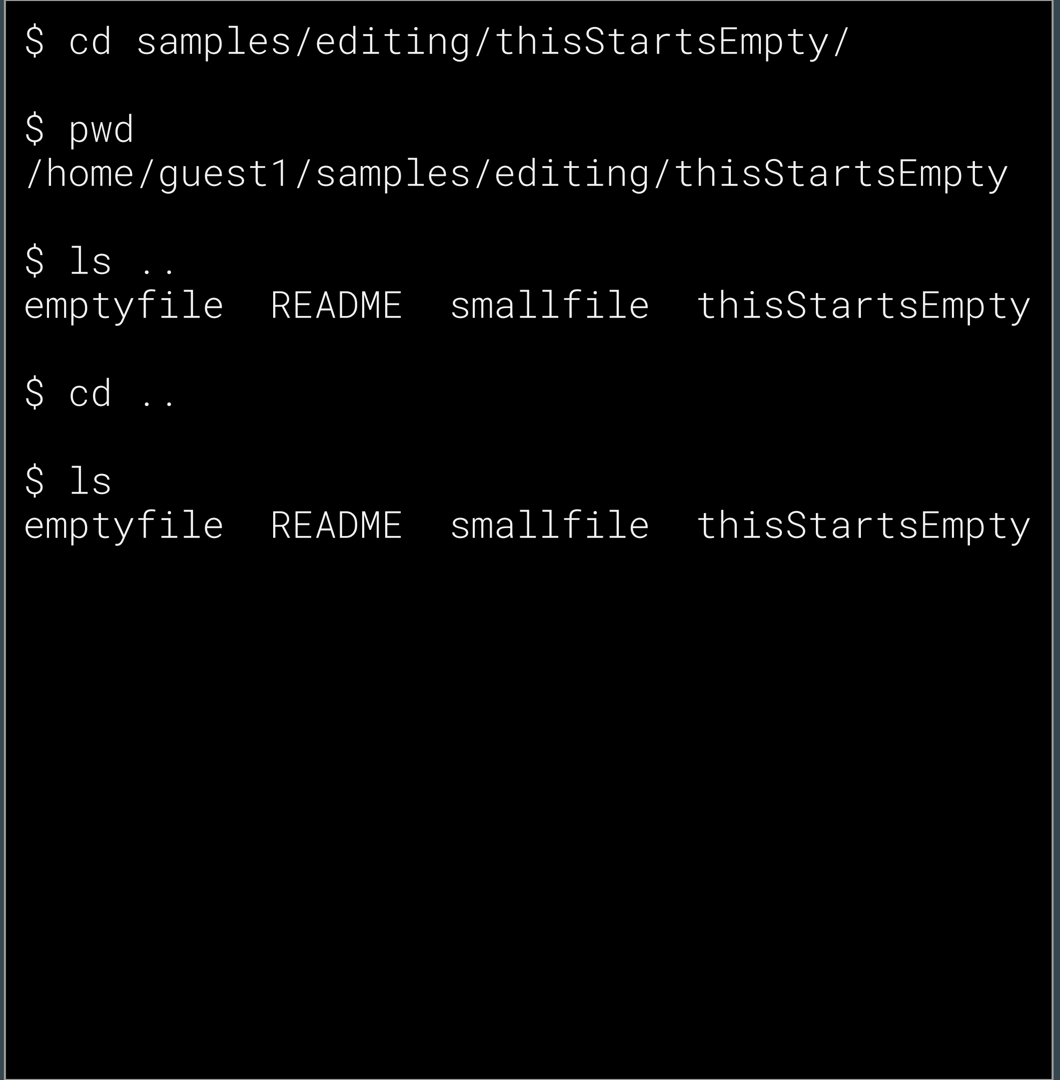
`cd`

Relative to your current or ‘working directory’ (used by many other programs including ls

. is short for this dir ./

.. is one directory up ../

~ is your home directory ~/



```
$ cd samples/editing/thisStartsEmpty/

$ pwd
/home/guest1/samples/editing/thisStartsEmpty

$ ls ..
emptyfile  README  smallfile  thisStartsEmpty

$ cd ..

$ ls
emptyfile  README  smallfile  thisStartsEmpty
```

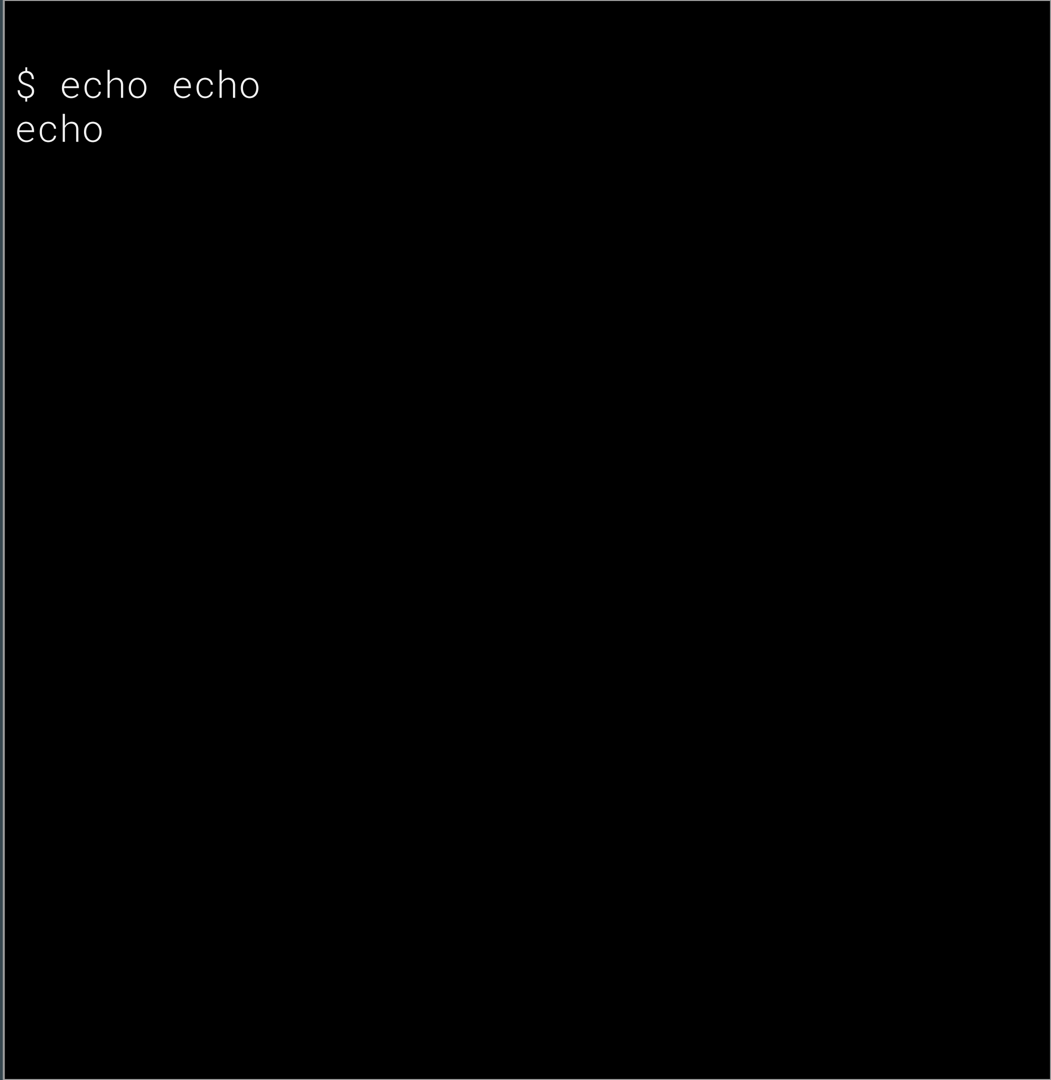
echo echo

echo <*words go here*>

echo

Display a line of text.

Returns to new line.



```
$ echo echo
echo
```

echolocation

echo <*words go here*>

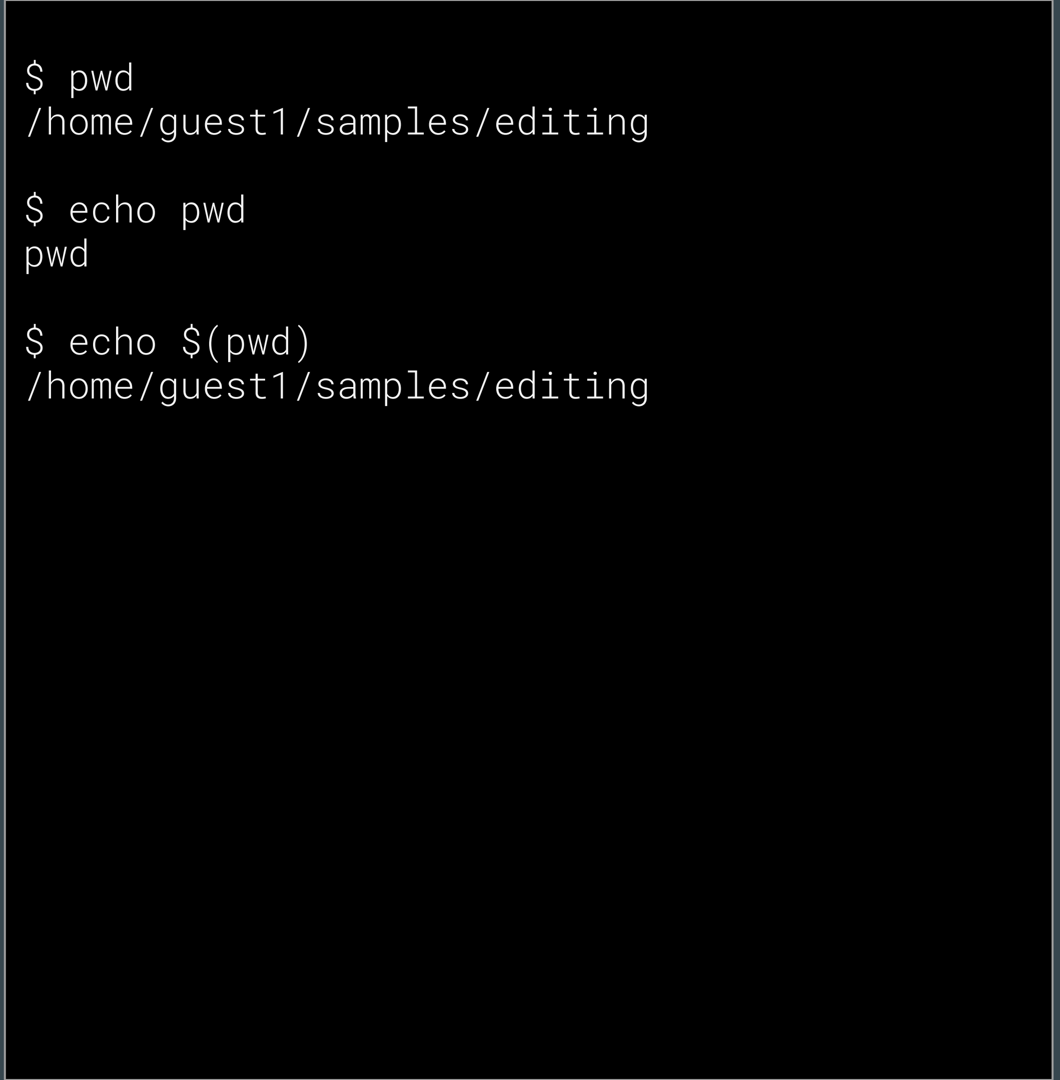
`echo $(pwd)`

A little more advanced:

$() runs the internal command, and replaces.

Like a composite function

echo(pwd)



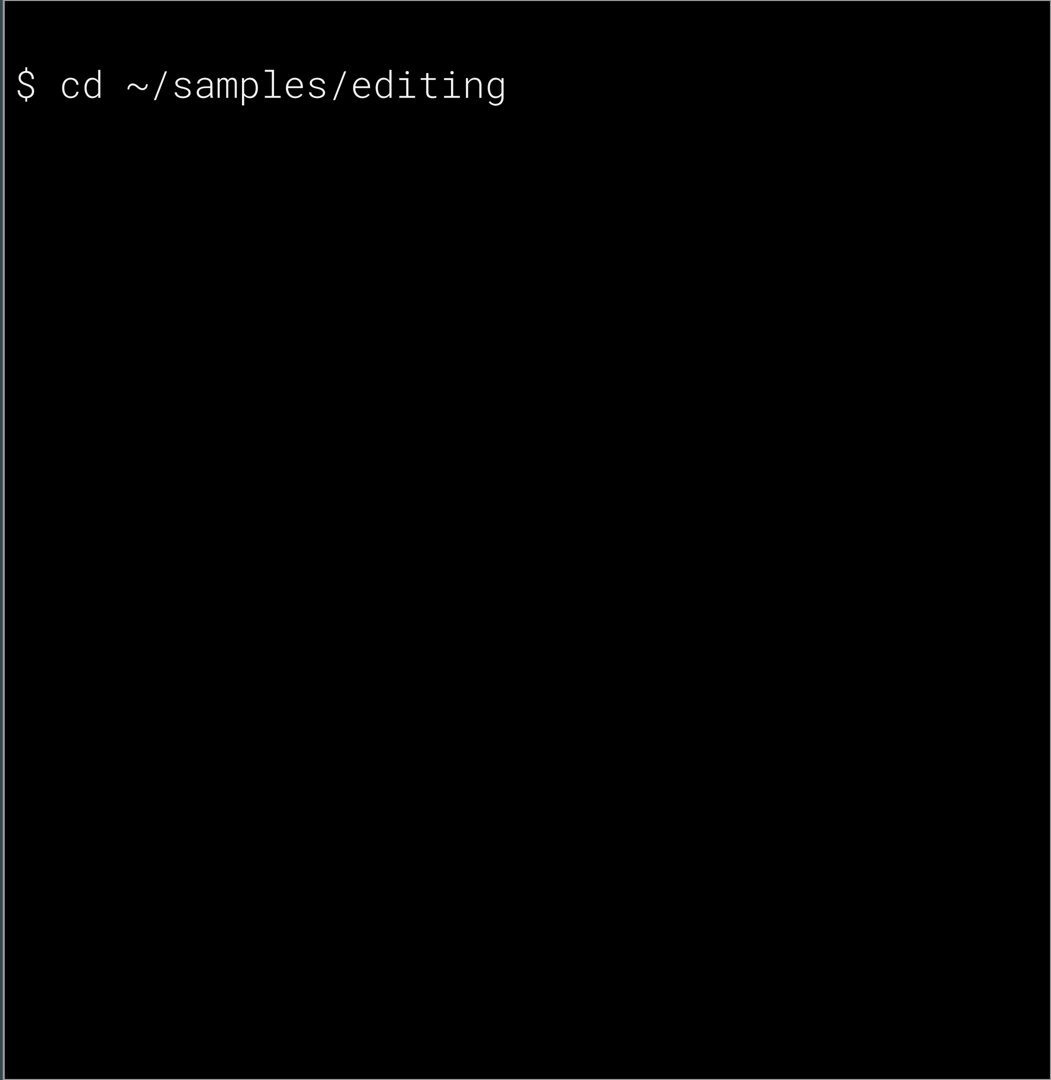
```
$ pwd
/home/guest1/samples/editing

$ echo pwd
pwd

$ echo $(pwd)
/home/guest1/samples/editing
```

Introducing the cat

cat <*file*>

`cat`

cat takes an input and copies it to the standard output.

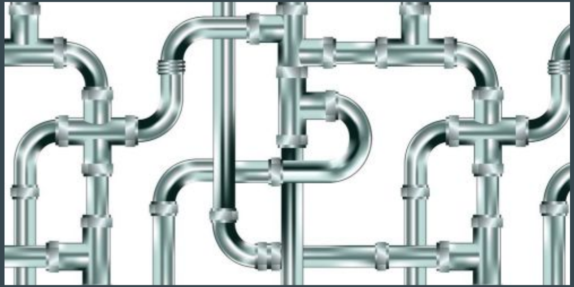


```
$ cd ~/samples/editing
```

What is Unix philosophy?

- Rule of Modularity: Write simple parts connected by clean interfaces.
- Rule of Clarity: Clarity is better than cleverness.
- Rule of Composition: Design programs to be connected to other programs.
- Rule of Simplicity: Design for simplicity; add complexity only where you must.
- Rule of Parsimony: Write a big program only when it is clear by demonstration that nothing else will do.
- Rule of Robustness: Robustness is the child of transparency and simplicity.
- Rule of Least Surprise: In interface design, always do the least surprising thing

This is a Pipe?



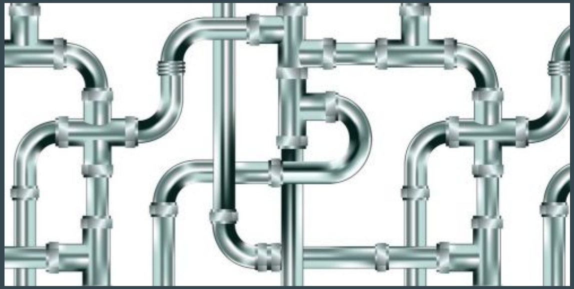
- Rule of Modularity: Write simple parts connected by clean interfaces.
- Rule of Clarity: Clarity is better than cleverness.
- Rule of Composition: Design programs to be connected to other programs.
- Rule of Simplicity: Design for simplicity; add complexity only where you must.

Pipes enable modularity, and composability.

And work around resource limitation and bottlenecks.

Act as connector for files and filelike objects as streams.

This is a Pipe?



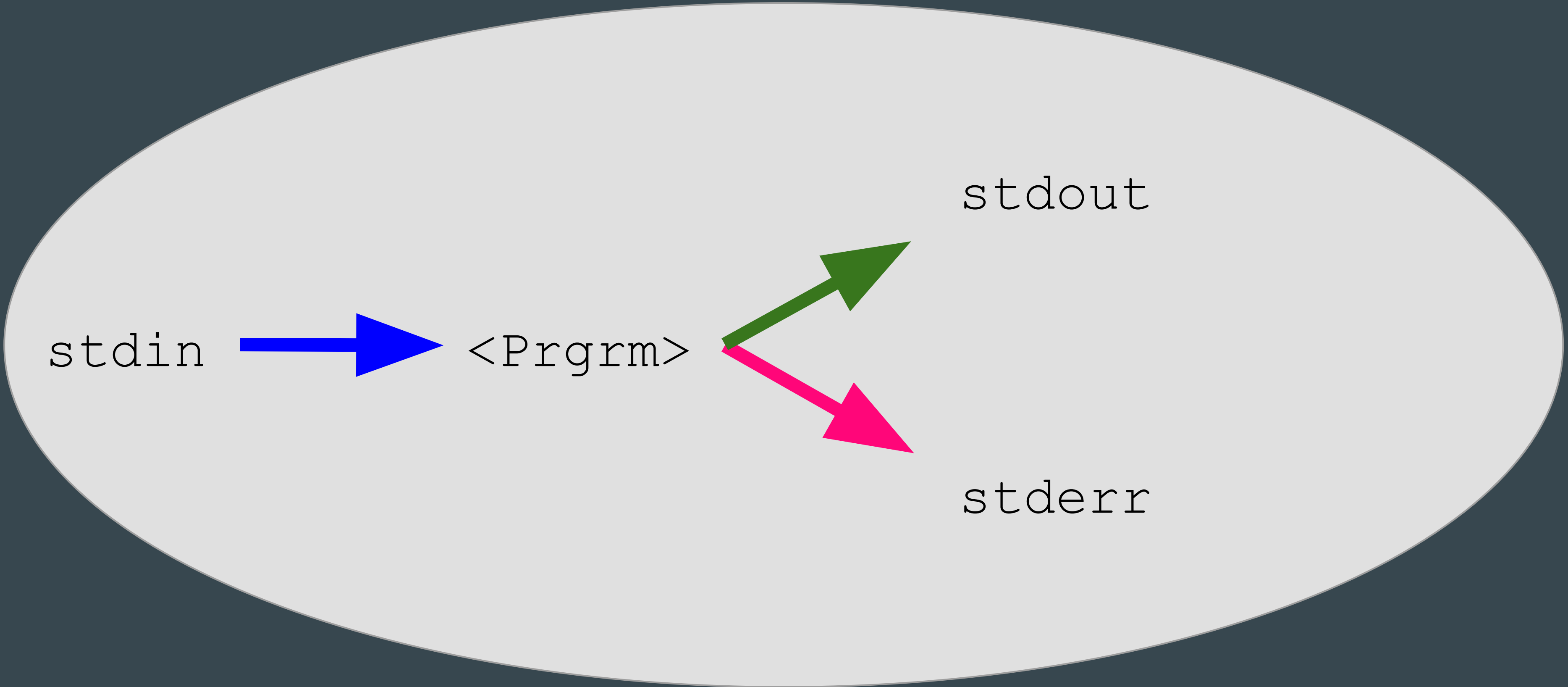
Act as connector for files and filelike objects as streams.

A stream presents the contents in the same way as when read from or written to a file.

This allows us to separate what goes where and re-direct it.

Redirection, integral to Unix philosophy

> \>\> | <



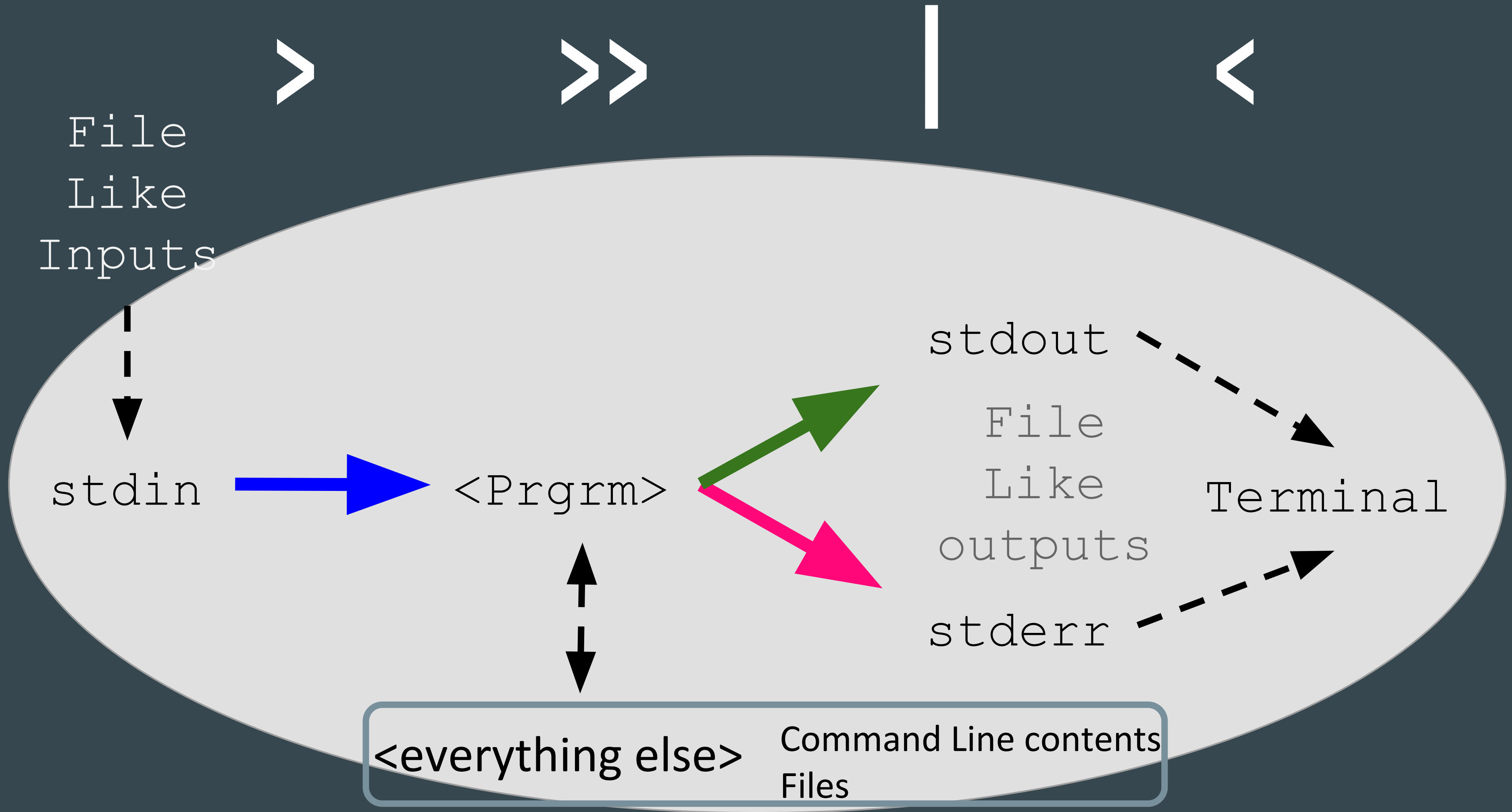
stdin

<Prgrm>

stdout

stderr

Redirection, integral to Unix philosophy



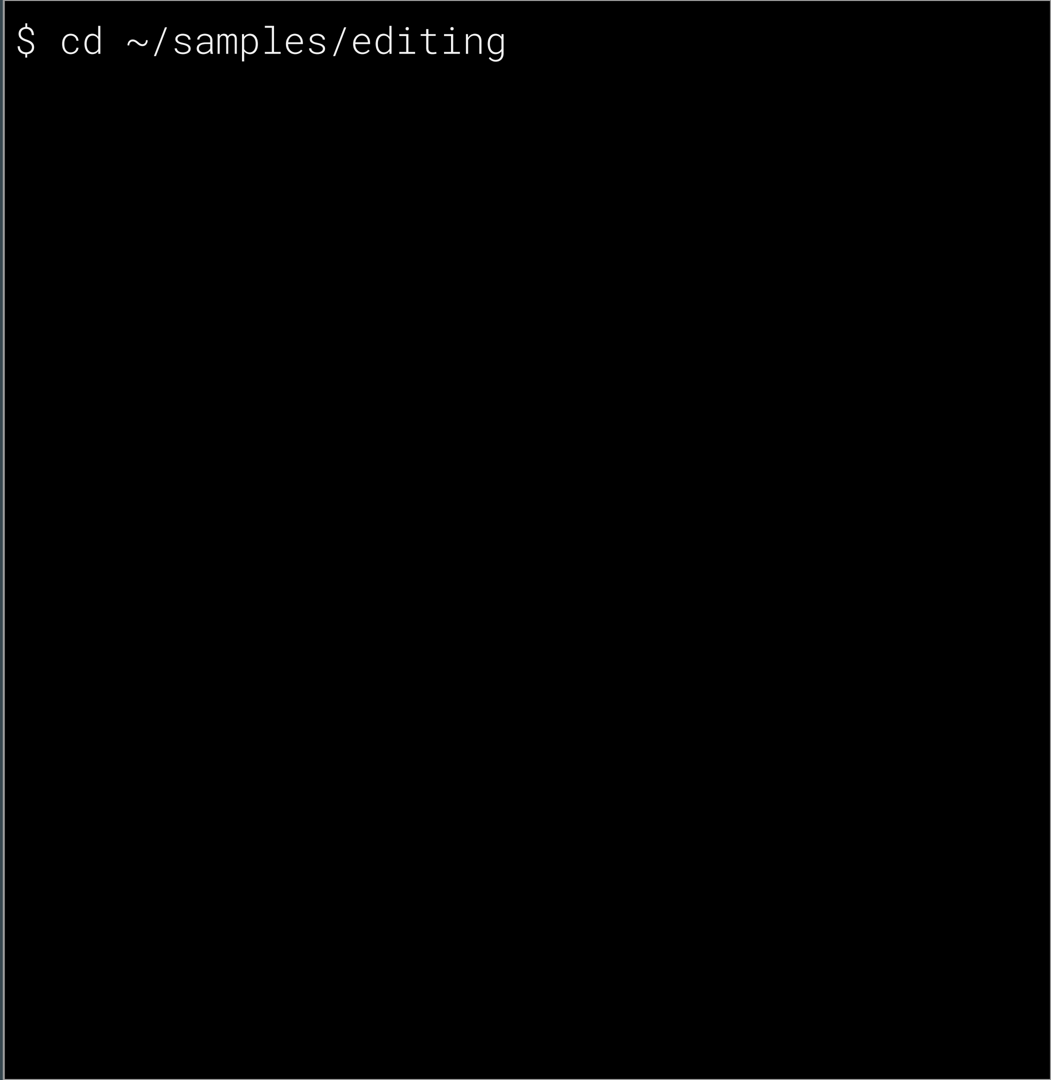
Clobbering time with >

echo words here > filename
cat filename

>

‘Right chevron’ or
‘Greater than’ sometimes
‘Pipe to file’

Opens a new file for writing, then writes the standard output to the file.



```
$ cd ~/samples/editing
```

Clobbering time with >

echo words here > filename
cat filename

>

‘Right chevron’ or
‘Greater than’ sometimes
‘Pipe to file’

Opens a new file for writing, then writes the standard output to the file.

```
$ echo sample words > samplewords.txt
$ cat samplewords.txt
sample words

$ echo other sample > samplewords.txt
$ cat samplewords.txt
other sample
```

Using your appendage >>

echo words here > filename
cat filename

>>

‘Append’ or ‘Append to file’

Open file,
go to end,
Then write standard output

```
$ echo sample words >> newsamplewords.txt
$ cat samplewords.txt
sample words

$ echo other sample >> newsamplewords.txt
$ cat samplewords.txt
sample words
other sample
```

Extraneous use of cat

cat < samplewords

<

Left chevron or less than,

Take the file on the right and uses it as the input for the command

```
$ cat < samplewords.txt
other words
```

Extraneous use of cat

cat -

For some commands the dash means take input from the command line

```
$ cat -
dafdas
dafdas
afda
afda
```

ls again

```
$ cd ~/samples/waspsample_J768

$ ls
J768_AHTWVVAFXX_ATGTCA_F7.1.fastq.gz           J768_AHV7WKAFXX_ATGTCA_F7.1.fastq.gz
J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed_fastqc.html
J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_fastqc.html
J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed.fq.gz
J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed.fq.gz
J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed_screen.png
J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_screen.png
```

ls again

```
$ cd ~/samples/waspsample_J768


$ ls -g
total 6087300
-rw-r--r--. 1 guests 1988764533 Oct 15 15:18 J768_AHTWVVAFXX_ATGTCA_F7.1.fastq.gz
-rw-r--r--. 1 guests 344983 Oct 15 15:18
J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed_fastqc.html
-rw-r--r--. 1 guests 1810943818 Oct 15 15:18
J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed.fq.gz
-rw-r--r--. 1 guests     6717 Oct 15 15:18
J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed_screen.png
-rw-r--r--. 1 guests 1272948542 Oct 15 15:18 J768_AHV7WKAFXX_ATGTCA_F7.1.fastq.gz
-rw-r--r--. 1 guests 340595 Oct 15 15:18
J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_fastqc.html
-rw-r--r--. 1 guests 1160020140 Oct 15 15:18
J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed.fq.gz
-rw-r--r--. 1 guests     6761 Oct 15 15:18
J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_screen.png
.1_trimmed_screen.png
```

ls again

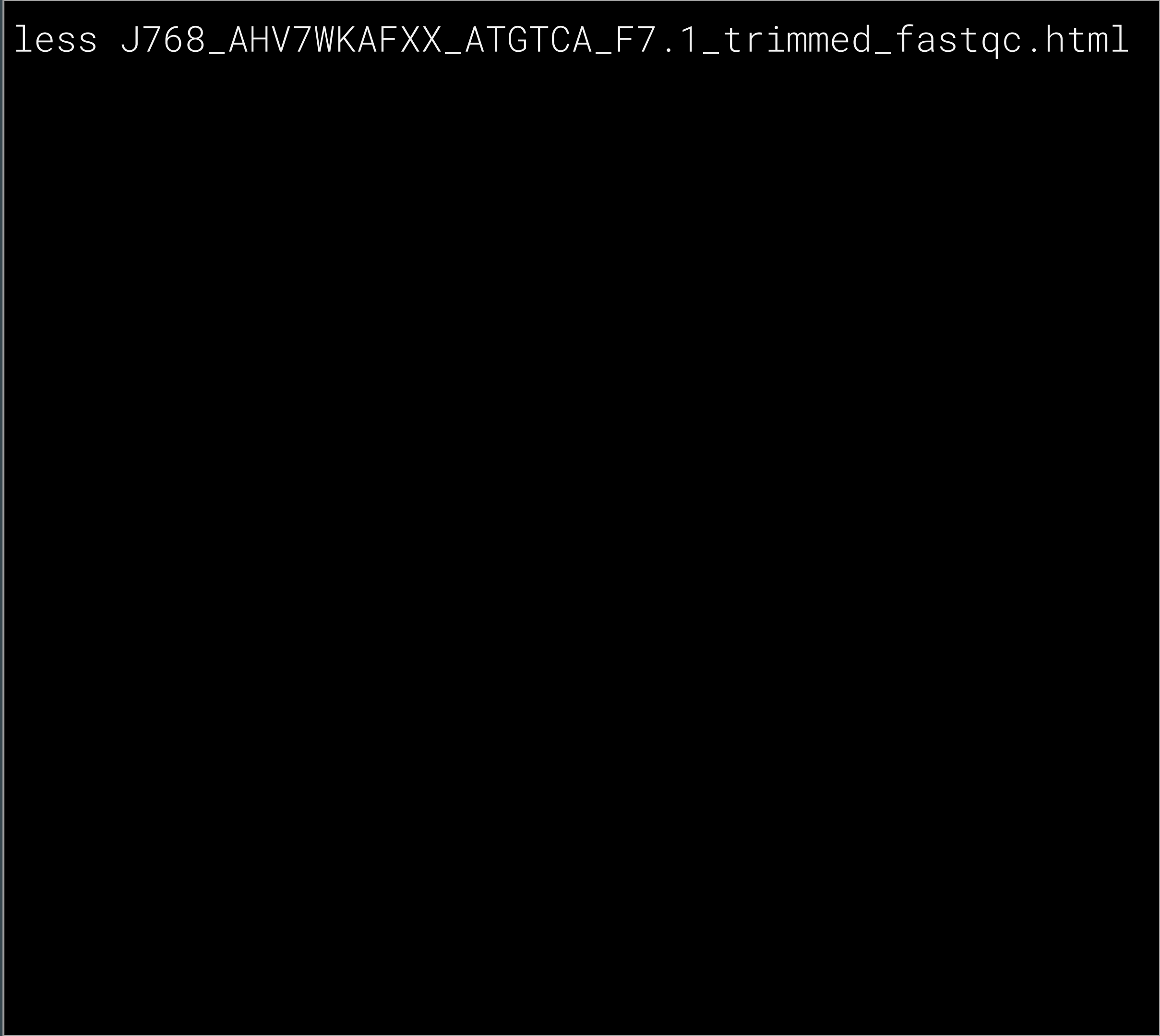
```
$ cd ~/samples/waspsample_J768


$ l-bash-4.2$ ls -gh
total 5.9G
-rw-r--r--. 1 guests 1.9G Oct 15 15:18 J768_AHTWVVAFXX_ATGTCA_F7.1.fastq.gz
-rw-r--r--. 1 guests 337K Oct 15 15:18
J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed_fastqc.html
-rw-r--r--. 1 guests 1.7G Oct 15 15:18 J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed.fq.gz
-rw-r--r--. 1 guests 6.6K Oct 15 15:18
J768_AHTWVVAFXX_ATGTCA_F7.1_trimmed_screen.png
-rw-r--r--. 1 guests 1.2G Oct 15 15:18 J768_AHV7WKAFXX_ATGTCA_F7.1.fastq.gz
-rw-r--r--. 1 guests 333K Oct 15 15:18
J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_fastqc.html
-rw-r--r--. 1 guests 1.1G Oct 15 15:18 J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed.fq.gz
-rw-r--r--. 1 guests 6.7K Oct 15 15:18 J768_AHV7WKAFXX_ATGTCA_F7
```

less

Less is a ‘pager’

You can move around with the up and down arrows

To leave press ‘q’



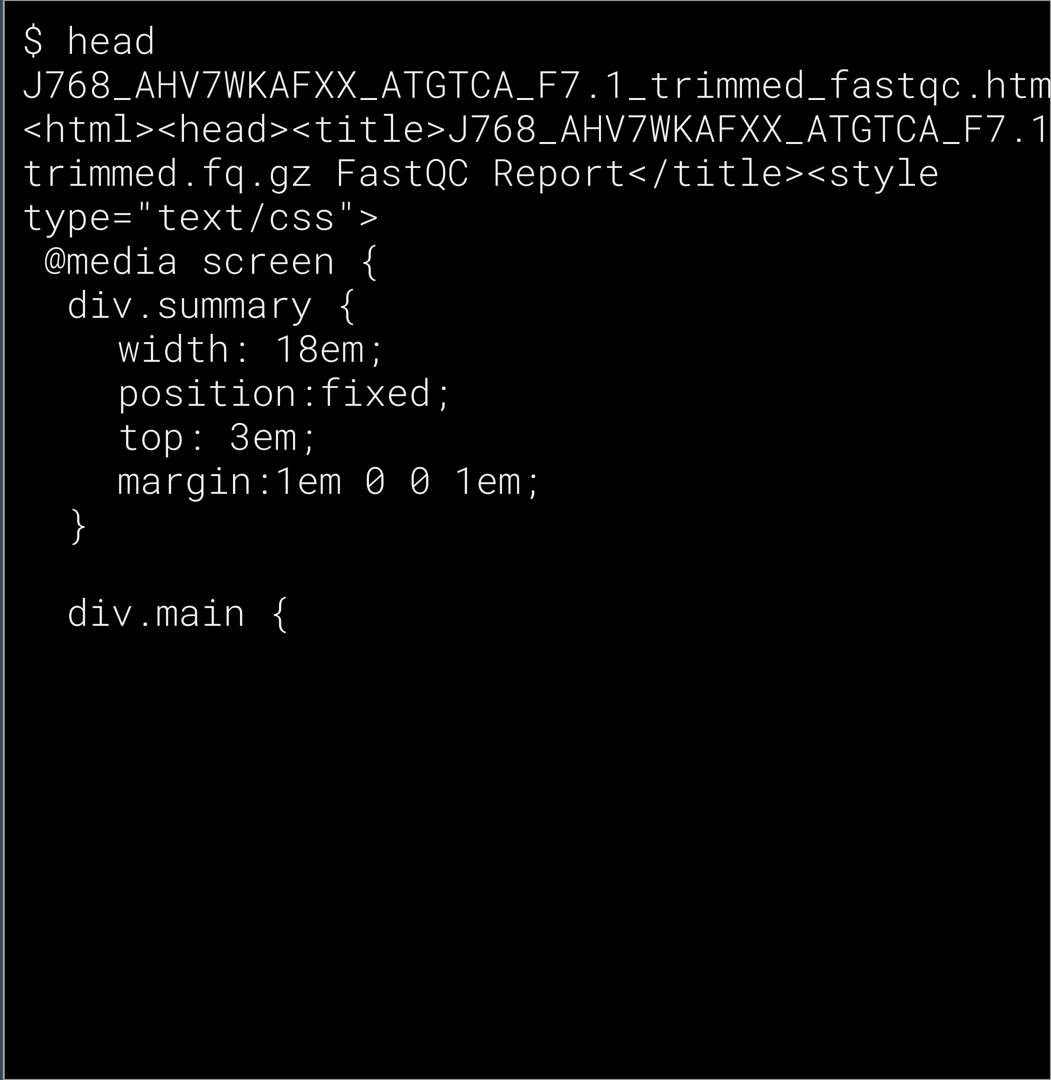
```
less J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_fastqc.html
```

Heads and tails

head file,

Head prints the first lines of a file.
10 by default, use

-n to adjust this



```
$ head
J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_fastqc.htm
<html><head><title>J768_AHV7WKAFXX_ATGTCA_F7.1
trimmed.fq.gz FastQC Report</title><style
type="text/css">
 @media screen {
  div.summary {
    width: 18em;
    position:fixed;
    top: 3em;
    margin:1em 0 0 1em;
  }

  div.main {
```

Heads and tails

tail file,

Like head except prints the last lines counting from the bottom

history

history prints a list of commands your used in order.

Put a number after it to limit it to the last few e.g.

```
history 5
```

Will only print the last 5 commands

History | less

Or pipe it to less or grep

```
$ history | less

$ history | grep tail
  131  tail
J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_fastqc.htm
  132  tail -n1
J768_AHV7WKAFXX_ATGTCA_F7.1_trimmed_fastqc.htm
  139  histoy | grep tail
  140  history | grep tail
```

! history

!

!12345

```
$ history | less

$ !
```

Make a new directory

mkdir <*dirname*>
rmdir <*empty-dir*>

mkdir   create a new directory

rmdir remove (delete) a directory

Only works if the directory is empty

```
$ cd

$ mkdir newdir

$ ls
newdir  samples
```

Make a new directory

mkdir <*dirname*>
rmdir <*empty-dir*>

mkdir create a new directory

rmdir remove (delete) a directory

Only works if the directory is empty

```
$ cd

$ mkdir newdir

$ ls
newdir  samples

$ rmdir newdir/

$ ls
samples

$ mkdir mydir
$ cd mydir
```

Make a new directory

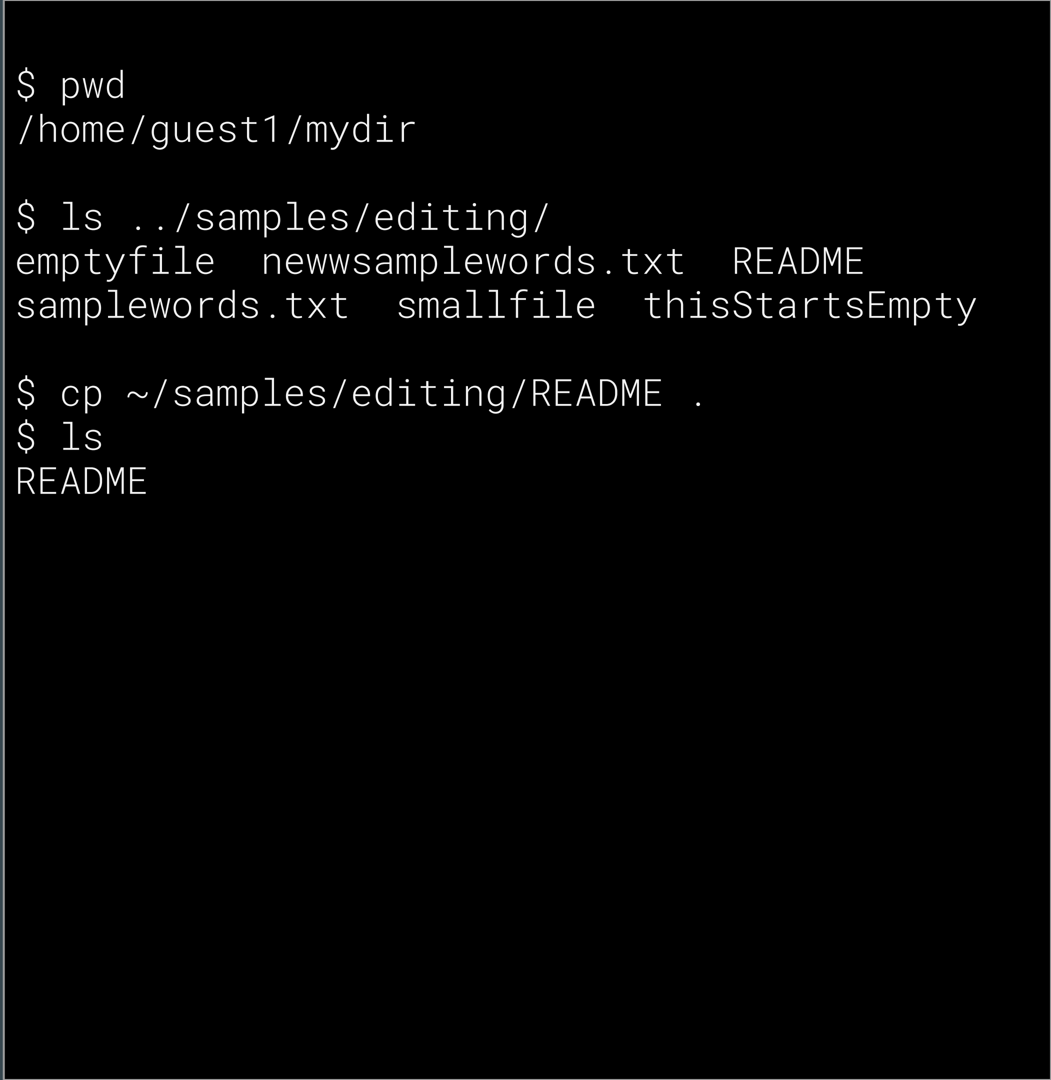
cp <file> <newfile>
mv <old> <new>

```
cp -R <dir> <newdir>
```

cp make a copy of old to new

cp -r   recursively copy

Copies a directory and all contents



```
$ pwd
/home/guest1/mydir

$ ls ../samples/editing/
emptyfile  newwsamplewords.txt  README
samplewords.txt  smallfile  thisStartsEmpty

$ cp ~/samples/editing/README .
$ ls
README
```

Make a new directory

cp <file> <newfile>
mv <old> <new>

```
cp -R <dir> <newdir>
```

cp make a copy of old to new

cp -r   recursively copy

Copies a directory and all contents

```
$ pwd
/home/guest1/mydir

$ ls ~/samples/editing/
emptyfile  newwsamplewords.txt  README
samplewords.txt  smallfile  thisStartsEmpty

$ cp ~/samples/editing/README .
$ ls
README

$ mv README README_copy
$ ls
README_copy
```

Make a new directory

rm
rm -r

`rm`

Use with caution - its permanent

```
$ cd ~/waspsample_J768
$ rm *.html
```

Editors and editor wars

Lots of possible editor

Vi, Vim . . . eventually you will use this it is on everything

Emacs . . . even more power - less popular now that everyone has a mouse

Nano (pico clone) almost a prevalent as vi, less power but easier to start on.

Quick vim

vi/m

To open a file
vim <*file*>

i to change to insert mode
Esc to leave

When not in insert mode
Core commands start with :

:w	save file with name used to open
:w filename	save as file name
:q	quite
:wq	save and quite

Quick nano

Like vim, but simpler

NO Modes, few addons

Instructions are easy to find

```
$nano testnano.txt
```

READING THE MANUAL

Places to look

- man
- info
- -h
- --help

How to know what to look up
apropos
man -k

Reading the manual for man

$ man man

Man is the manuals program

- man man
    - Built in manual on the manual
    - Try:
        - man ls
        - man cat
        - man grep
            - Search with key matching or regular expressions
        - man bash
            - This is the shell language we’ve been using

$ man -k or $ apropos to do a word search of the manuals

```
$ man -k manual
apropos (1)          - search the manual page names and descriptions
catman (8)           - create or update the pre-formatted manual pages
gtar (1)             - manual page for tar 1.26
man (1)              - an interface to the on-line reference manuals
manconv (1)          - convert manual page from one encoding to another
mandb (8)            - create or update the manual page index caches
manpath (1)          - determine search path for manual pages
tar (1)        - manual page for tar 1.26
whatis (1)           - display manual page descriptions
whereis (1)          - locate the binary, source, and manual page files for a command
```

Reading the manual for man

Keyword searching in less

- Many programs like man, use less to show you pages of information
- Set by an environment variable see how our system is doing it with:
  - printenv | grep PATH
  - printenv | less

Converting from microsoft

Cat, dos2unix

Different operating systems use non-printing characters differently.
(this can be a headache)

See them with cat
$ cat -v or $ cat -A

Convert with dos2unix and unix2dos

printenv, export, alias

- Many programs like man, use less to show you pages of information
- Set by an environment variable see how our system is doing it with:
  - printenv | grep less

Other useful links

- http://homepages.uc.edu/~thomam/Intro_Unix_Text/TOC.html
- https://ryanstutorials.net/linuxtutorial/piping.php
- http://www.linfo.org/pipes.html
- https://brandonwamboldt.ca/how-linux-pipes-work-under-the-hood-1518/
- https://www.cs.rutgers.edu/~pxk/416/notes/c-tutorials/pipe.html