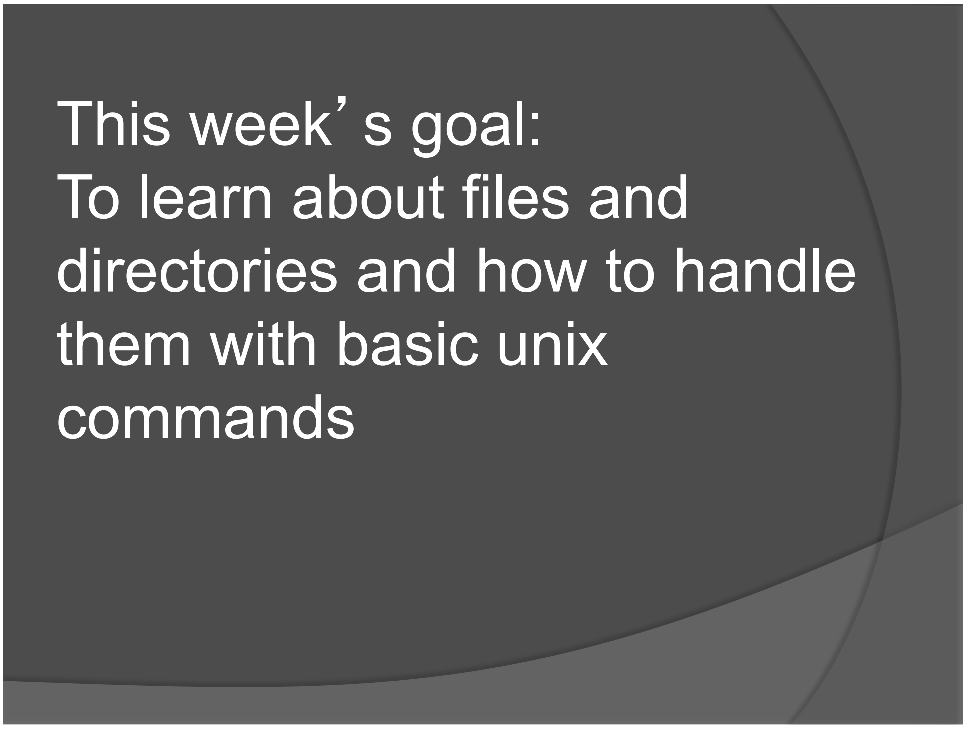
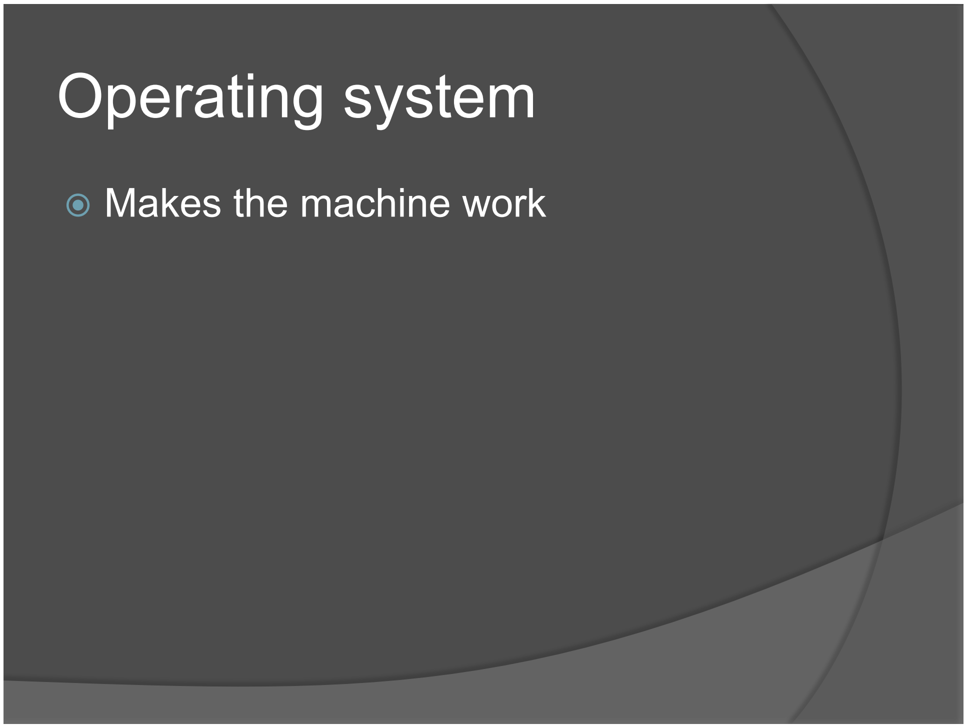
BASIC UNIX SKILLS

BINF 511, Winter 2017

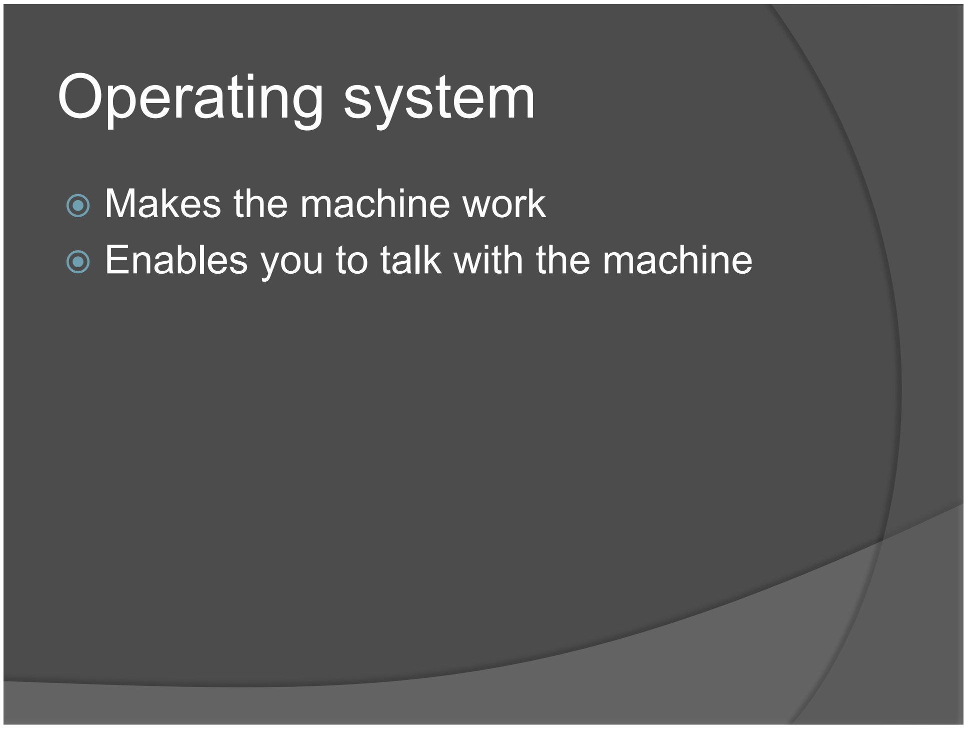


This week’s goal:
To learn about files and directories and how to handle them with basic unix commands



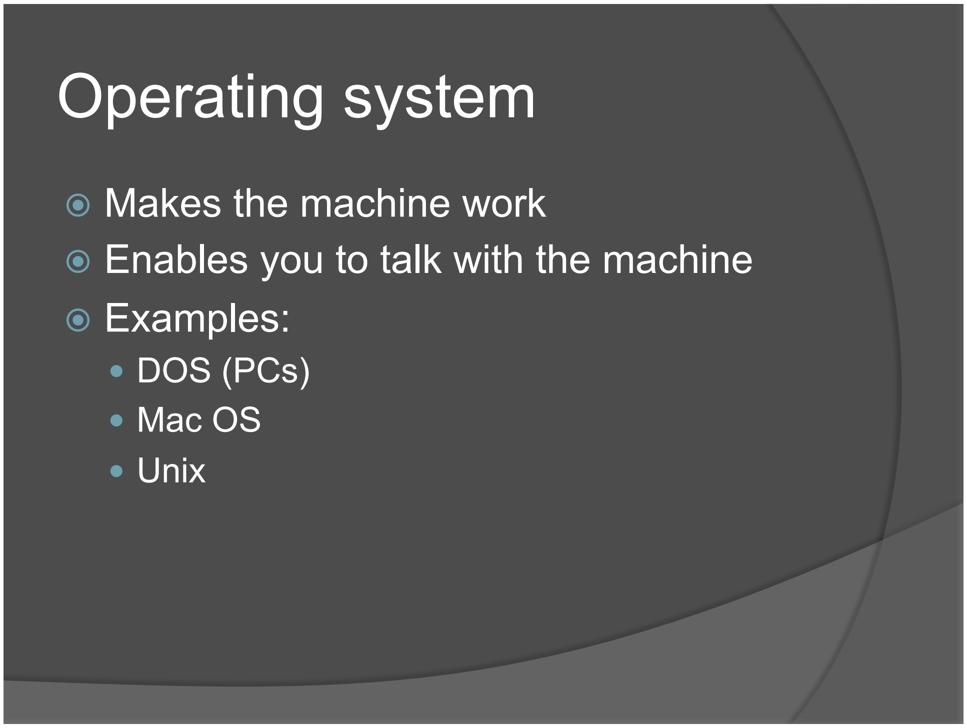
Operating system

- Makes the machine work



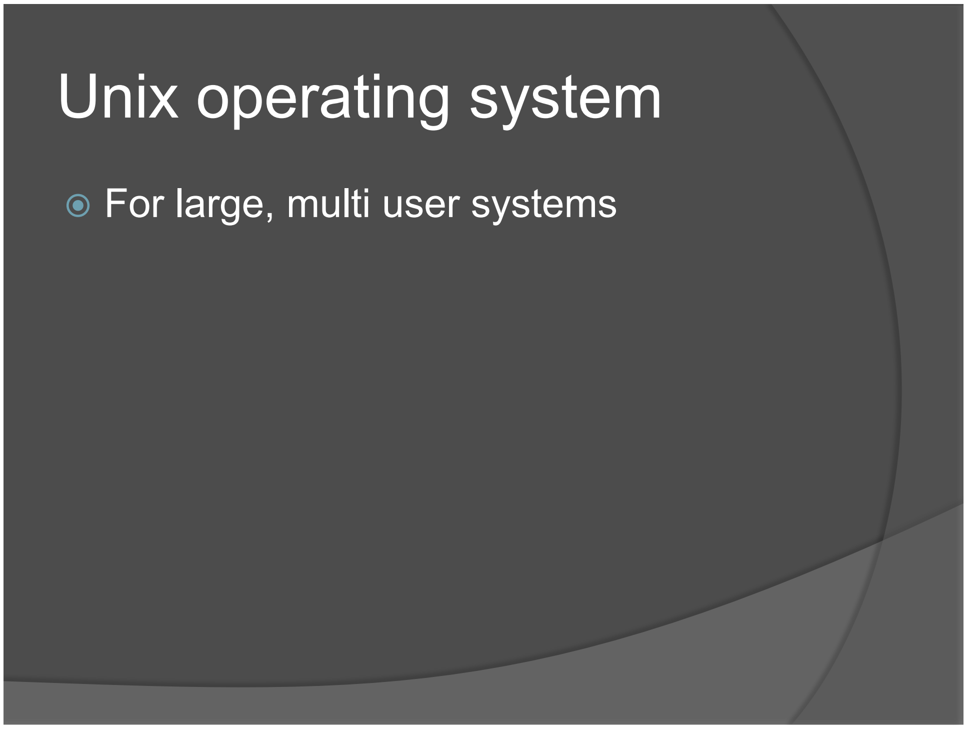
Operating system

- Makes the machine work
- Enables you to talk with the machine



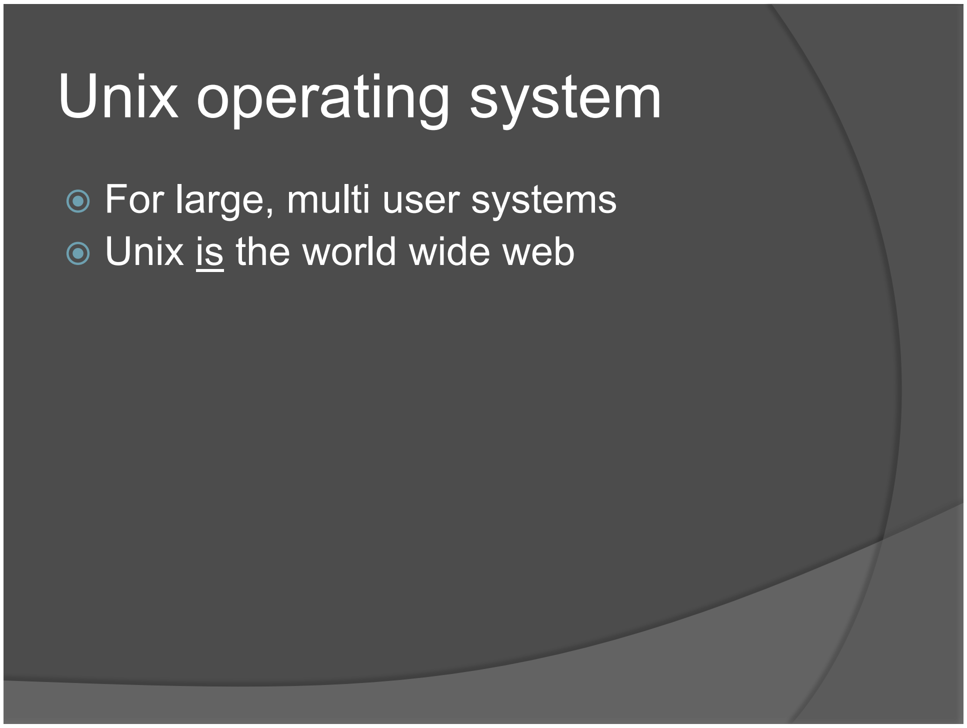
Operating system

- Makes the machine work
- Enables you to talk with the machine
- Examples:
  - DOS (PCs)
  - Mac OS
  - Unix



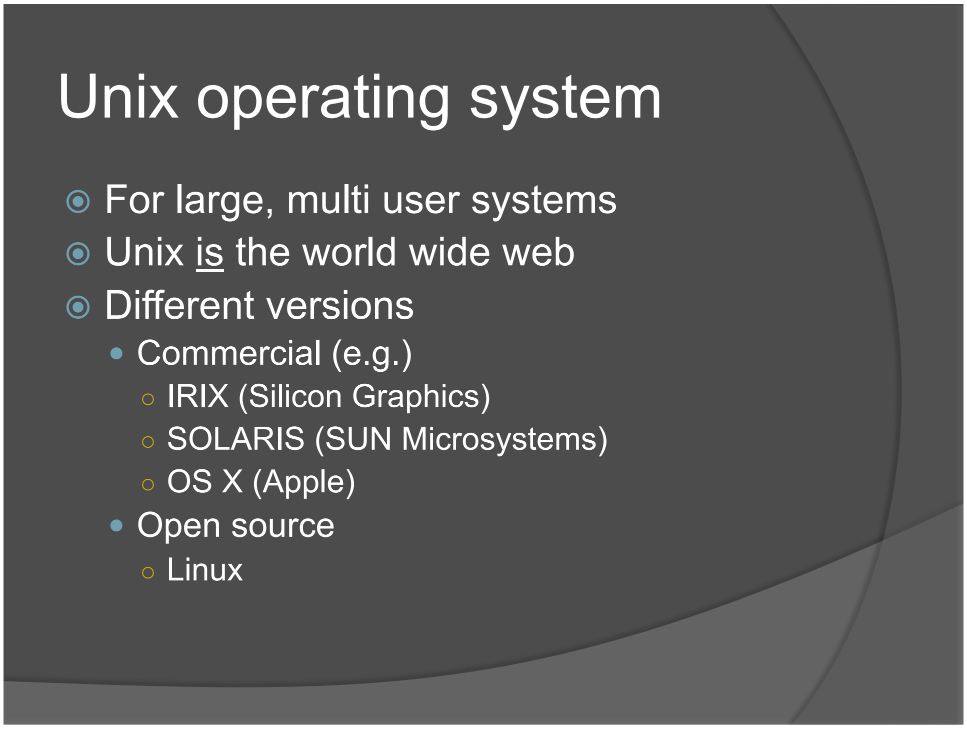
Unix operating system

- For large, multi user systems



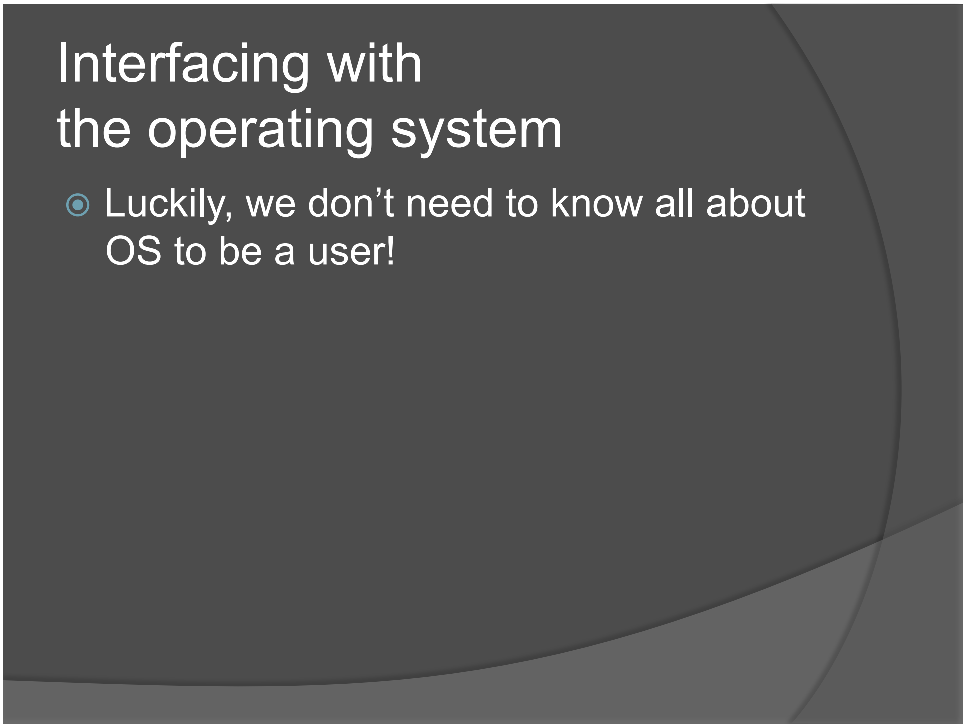
Unix operating system

- For large, multi user systems
- Unix is the world wide web



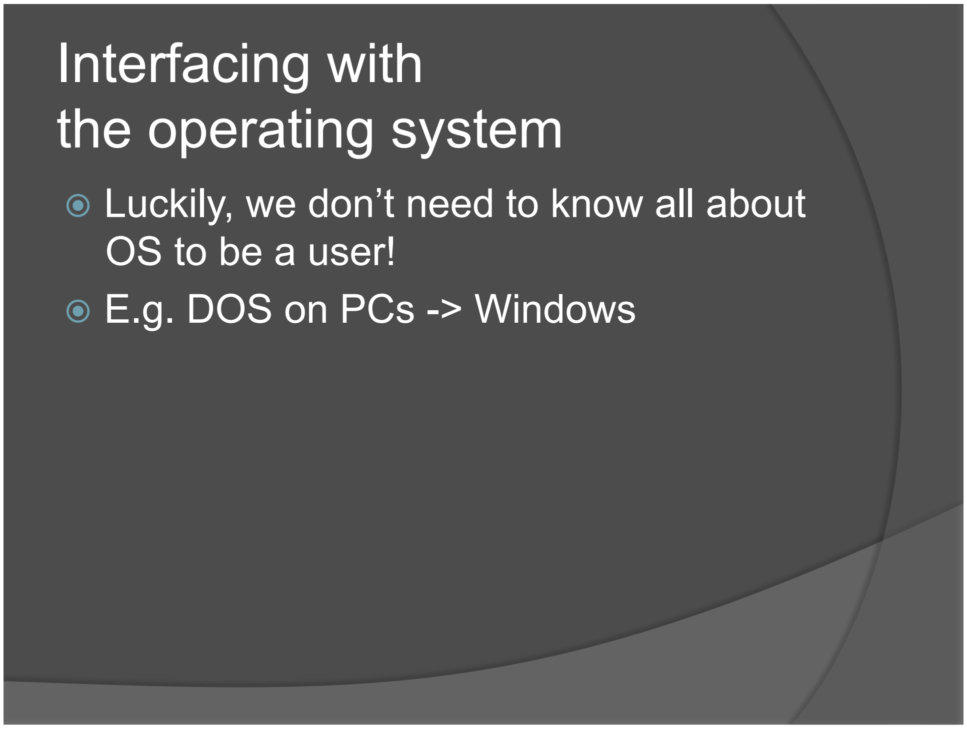
Unix operating system

- For large, multi user systems
- Unix is the world wide web
- Different versions
  - Commercial (e.g.)
    - IRIX (Silicon Graphics)
    - SOLARIS (SUN Microsystems)
    - OS X (Apple)
  - Open source
    - Linux



Interfacing with the operating system

- Luckily, we don’t need to know all about OS to be a user!



Interfacing with the operating system

- Luckily, we don’t need to know all about OS to be a user!
- E.g. DOS on PCs -> Windows

Interfacing with the operating system

- Luckily, we don’t need to know all about OS to be a user!
- E.g. DOS on PCs -> Windows
- Unix systems use X-window
  - You can have many windows open at once

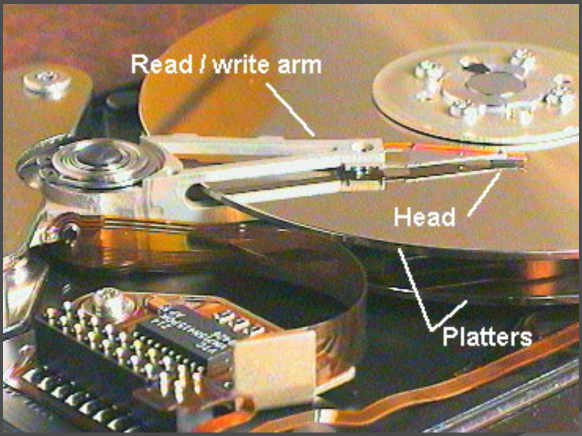
(File, don’t pile!)

FILE SYSTEMS BASICS

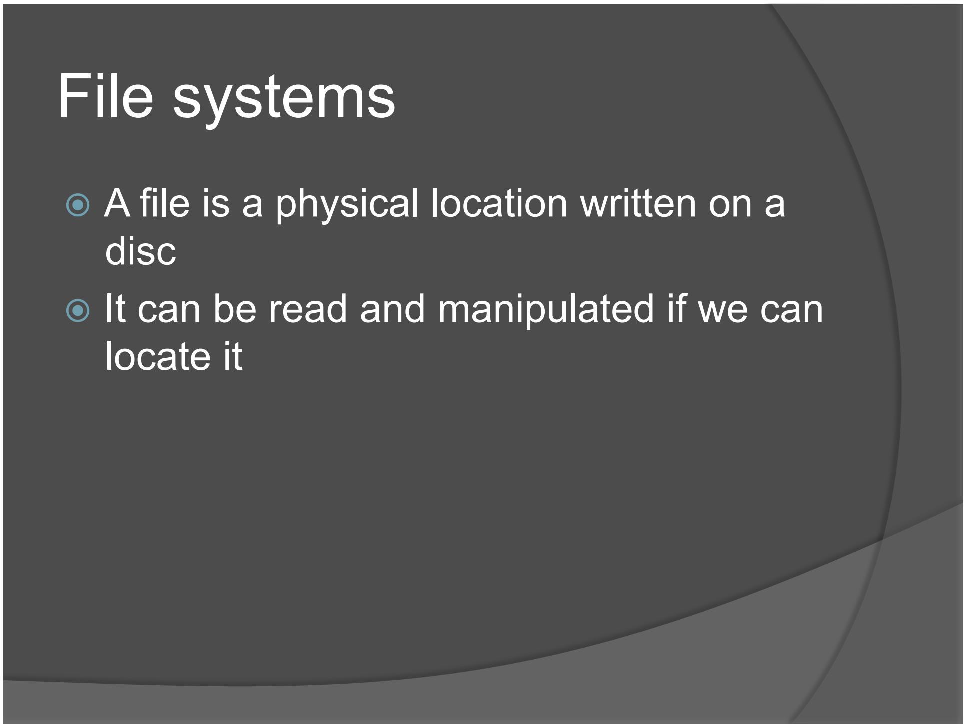
File systems

- A file is a physical location written on a disc

Hard disc

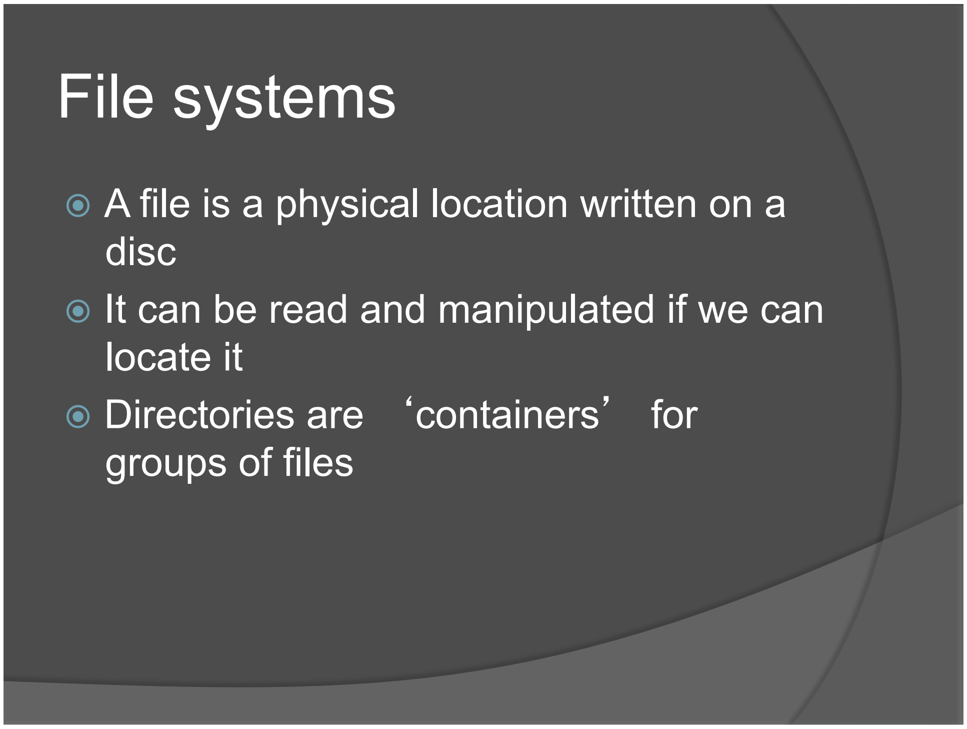


http://www.microscopy-uk.net/mag/indexmag.html



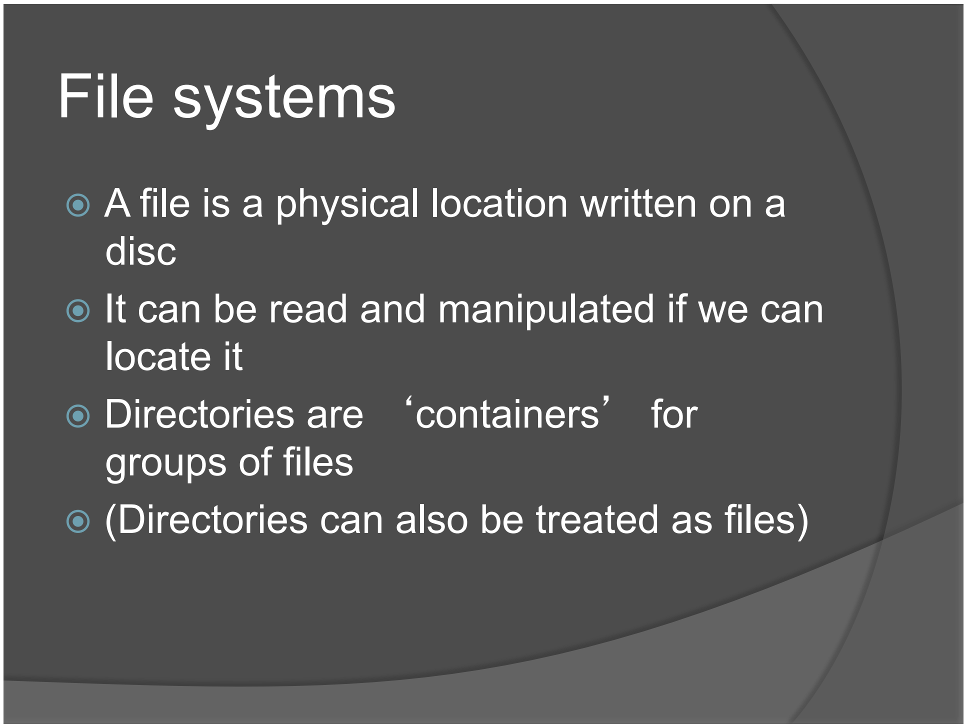
File systems

- A file is a physical location written on a disc
- It can be read and manipulated if we can locate it



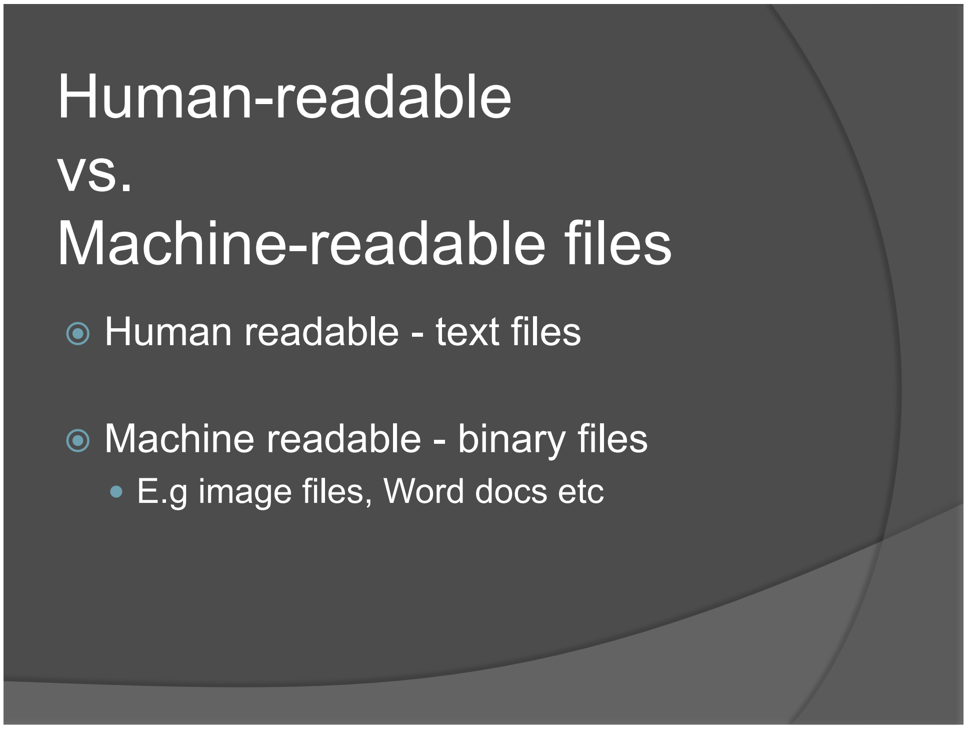
File systems

- A file is a physical location written on a disc
- It can be read and manipulated if we can locate it
- Directories are ‘containers’ for groups of files



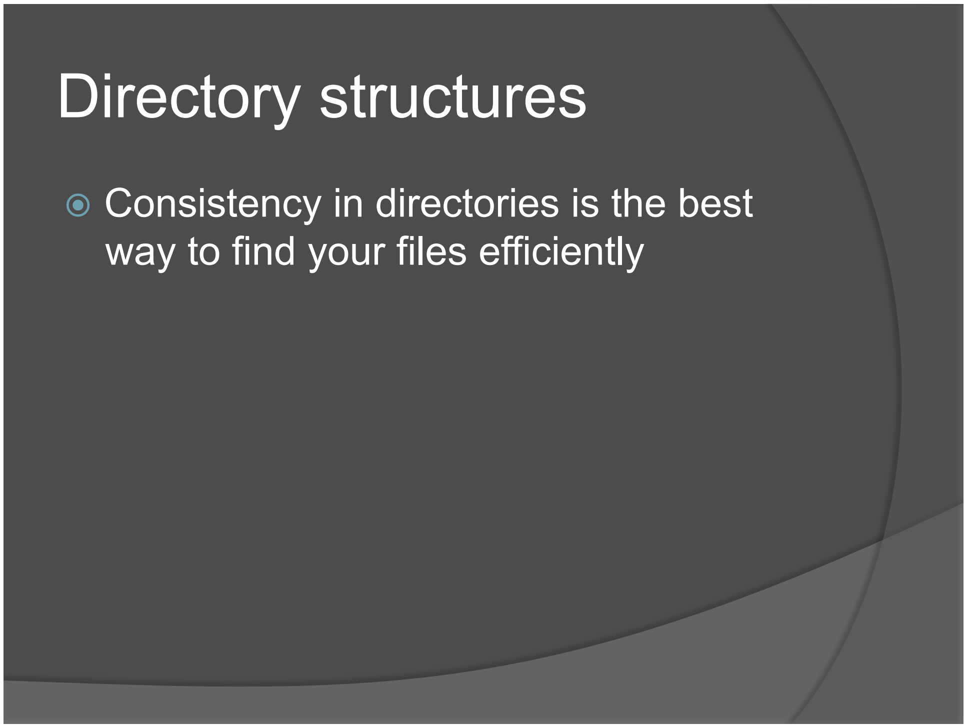
File systems

- A file is a physical location written on a disc
- It can be read and manipulated if we can locate it
- Directories are ‘containers’ for groups of files
- (Directories can also be treated as files)



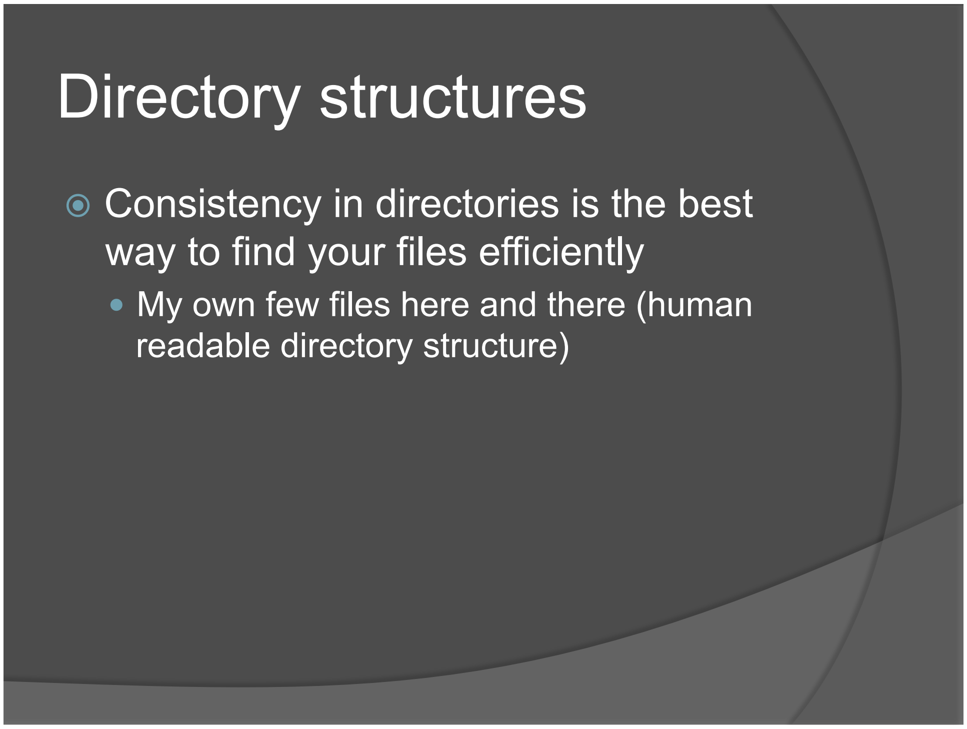
Human-readable vs. Machine-readable files

- Human readable - text files
- Machine readable - binary files
  - E.g image files, Word docs etc



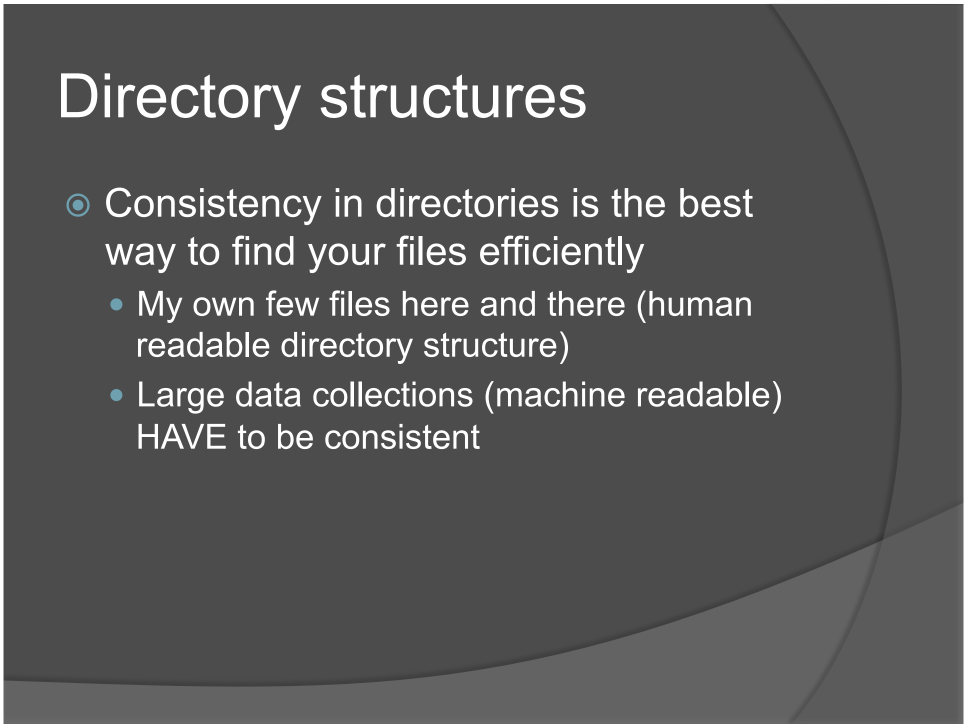
Directory structures

- Consistency in directories is the best way to find your files efficiently



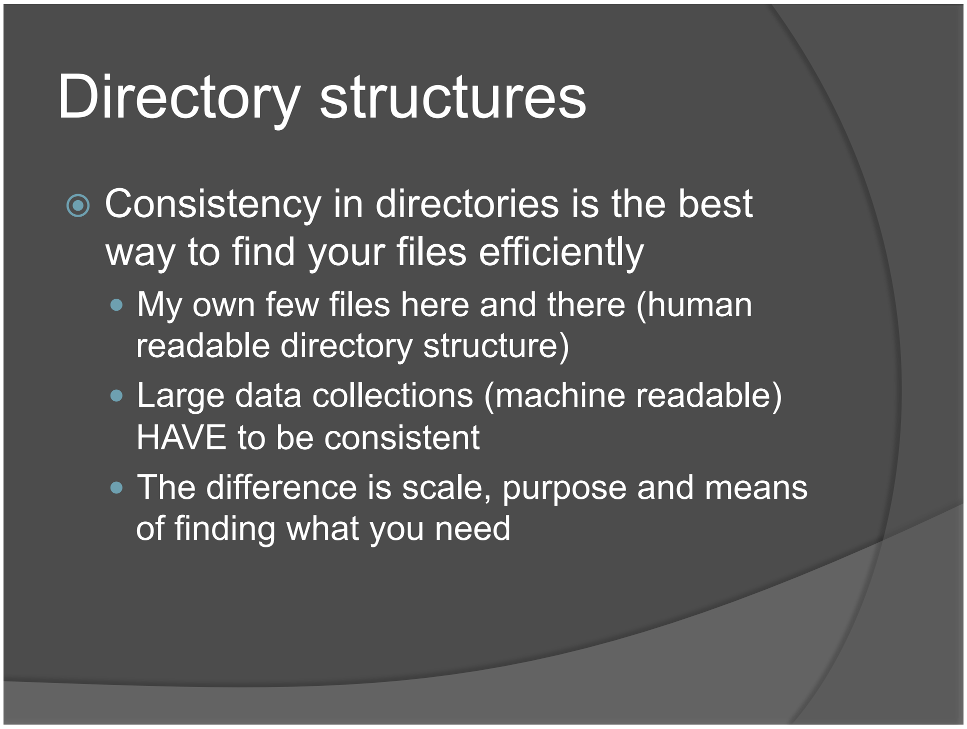
Directory structures

- Consistency in directories is the best way to find your files efficiently
  - My own few files here and there (human readable directory structure)



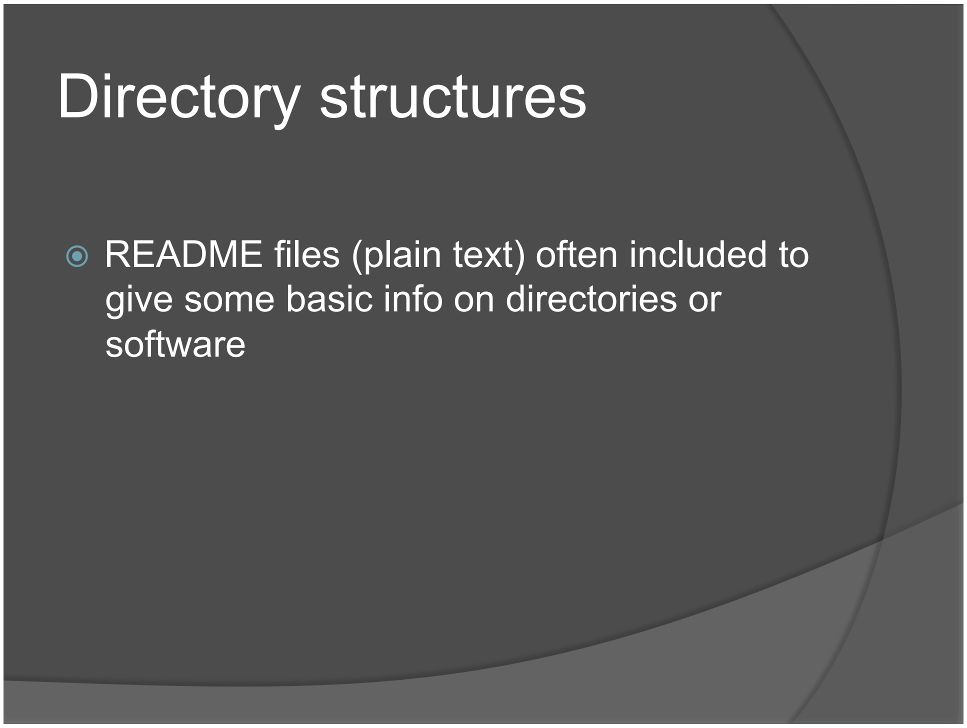
Directory structures

- Consistency in directories is the best way to find your files efficiently
  - My own few files here and there (human readable directory structure)
  - Large data collections (machine readable) HAVE to be consistent



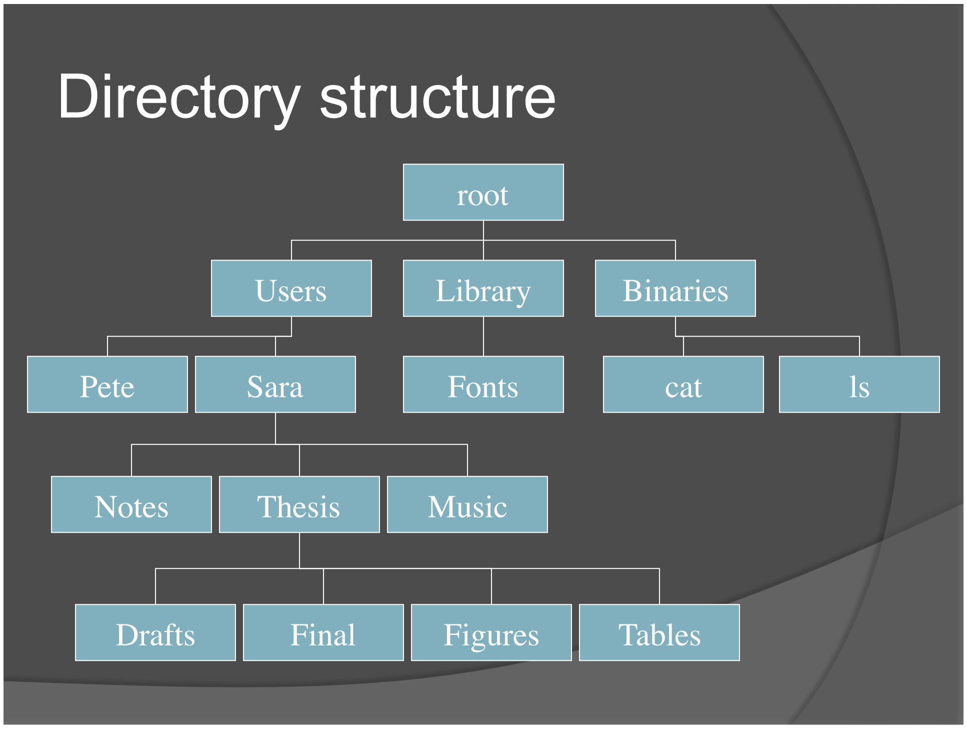
Directory structures

- Consistency in directories is the best way to find your files efficiently
  - My own few files here and there (human readable directory structure)
  - Large data collections (machine readable) HAVE to be consistent
  - The difference is scale, purpose and means of finding what you need



Directory structures

- README files (plain text) often included to give some basic info on directories or software



Directory structure

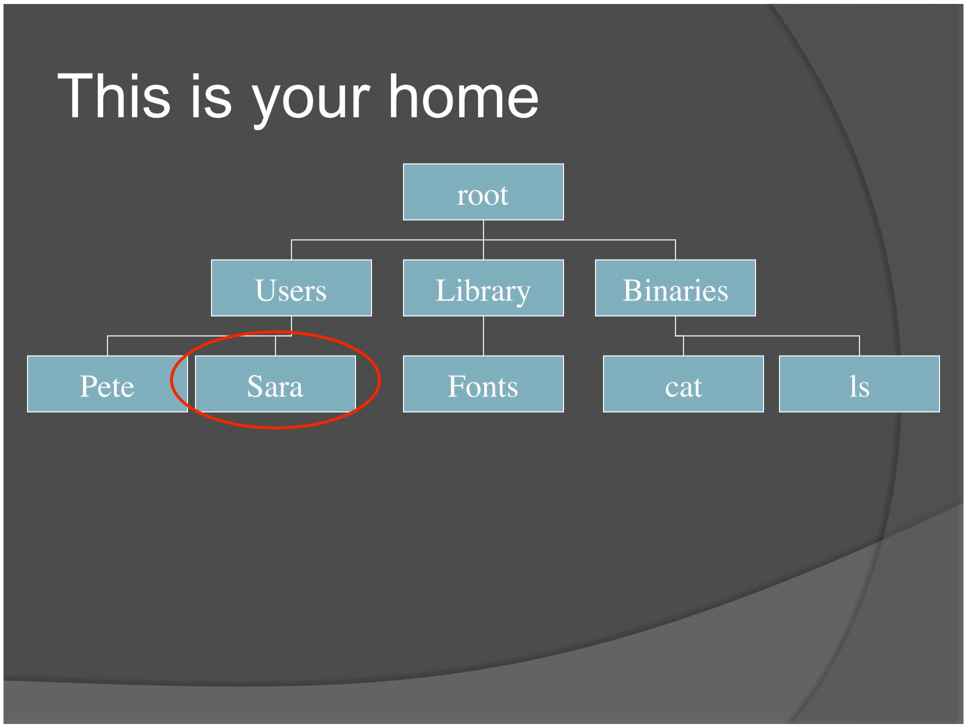
root

Users Library Binaries

Pete Sara Fonts cat ls

Notes Thesis Music

Drafts Final Figures Tables

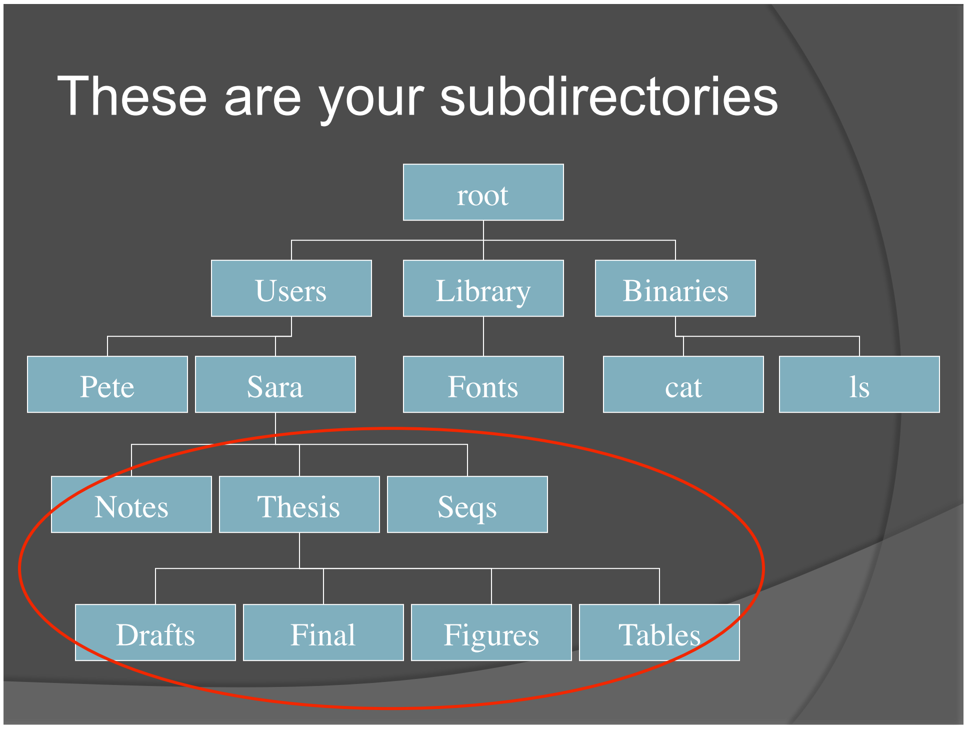


This is your home

root

Users Library Binaries

Pete Sara Fonts cat ls



These are your subdirectories

root

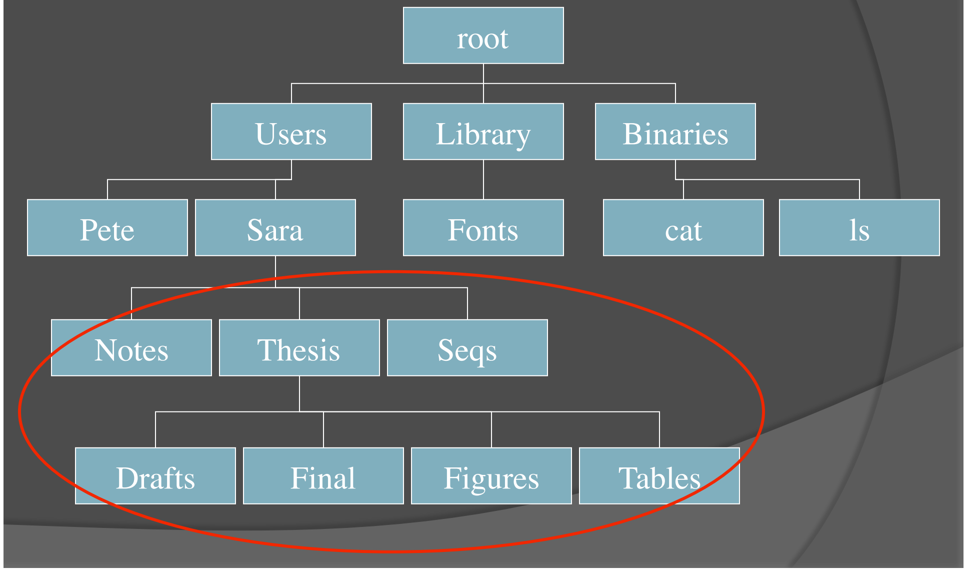
Users Library Binaries

Pete Sara Fonts cat ls

Notes Thesis Seqs

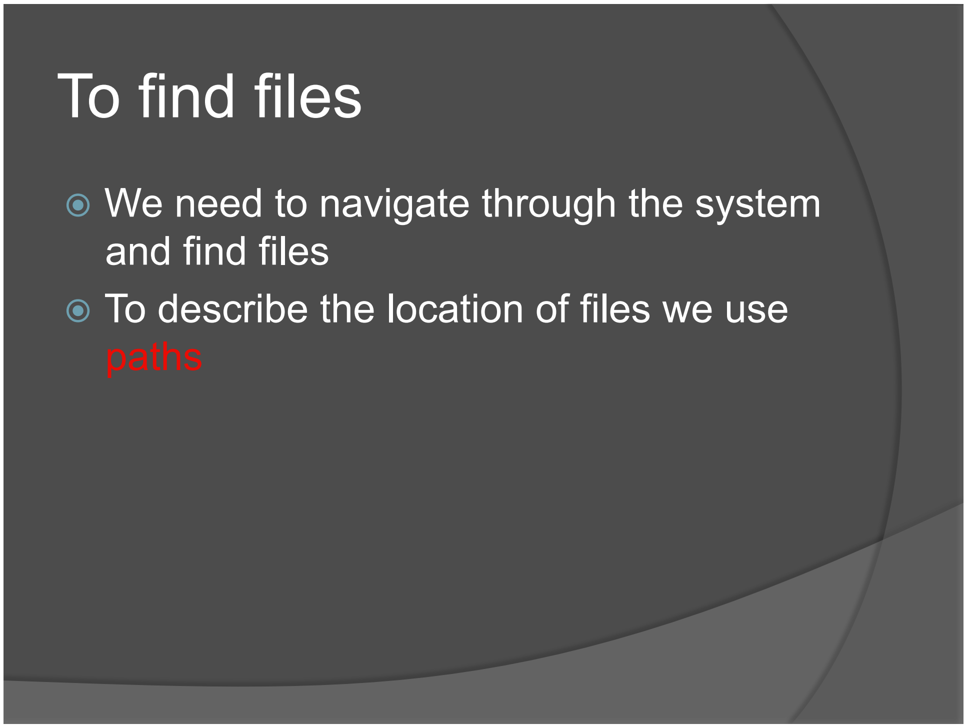
Drafts Final Figures Tables

“Sara” is the parent of those subdirectories



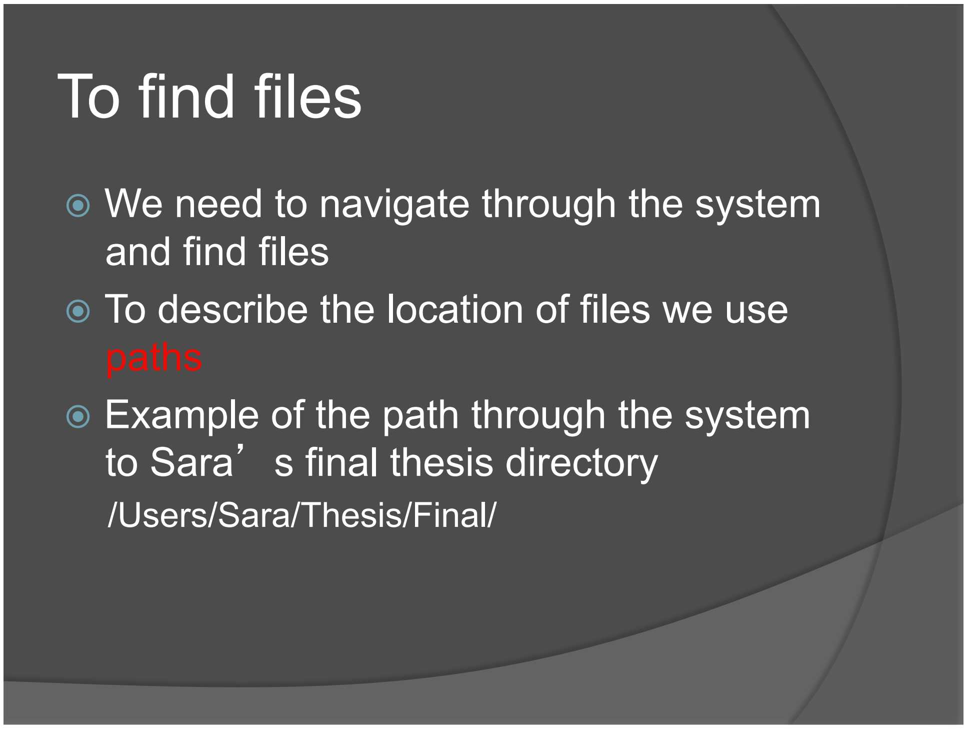
To find files

- We need to navigate through the system and find files



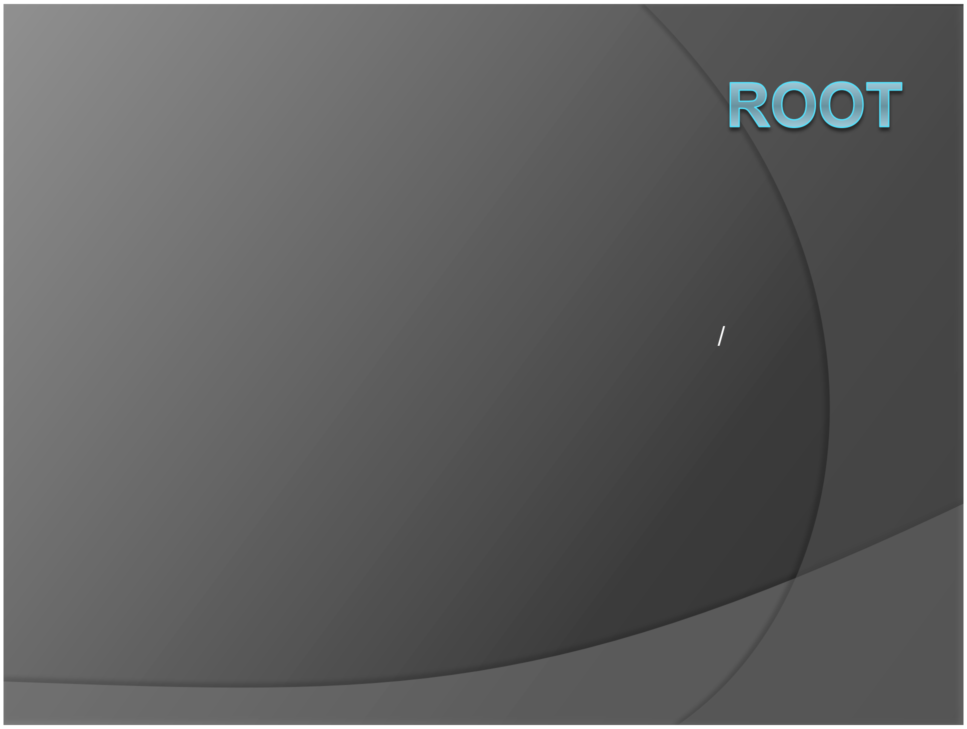
To find files

- We need to navigate through the system and find files
- To describe the location of files we use paths



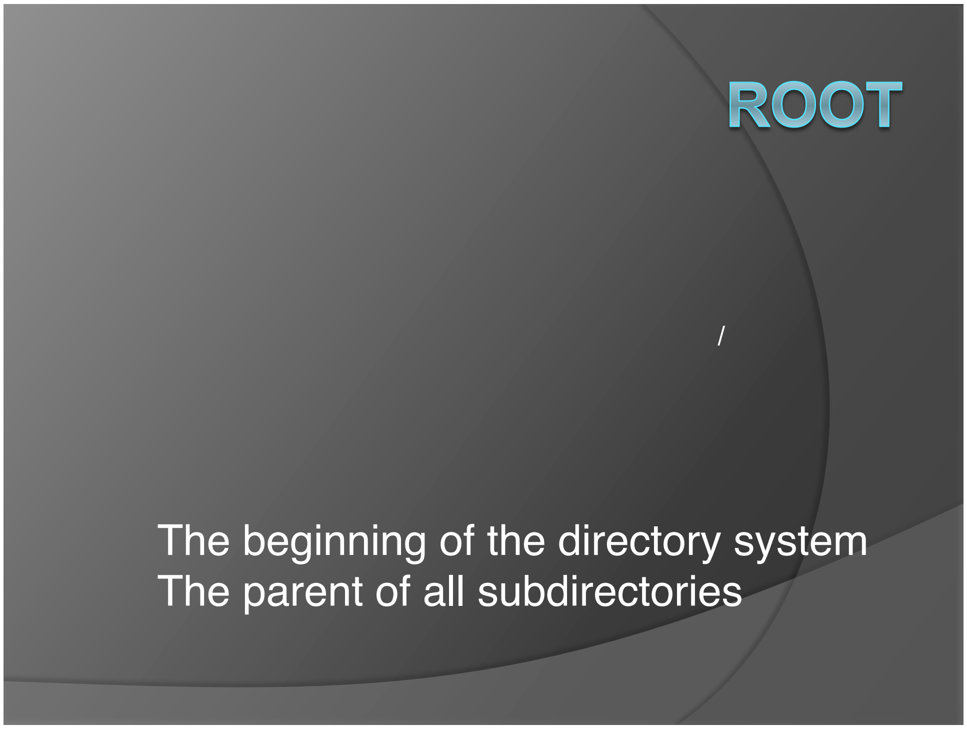
To find files

- We need to navigate through the system and find files
- To describe the location of files we use paths
- Example of the path through the system to Sara' s final thesis directory /Users/Sara/Thesis/Final/



ROOT

/



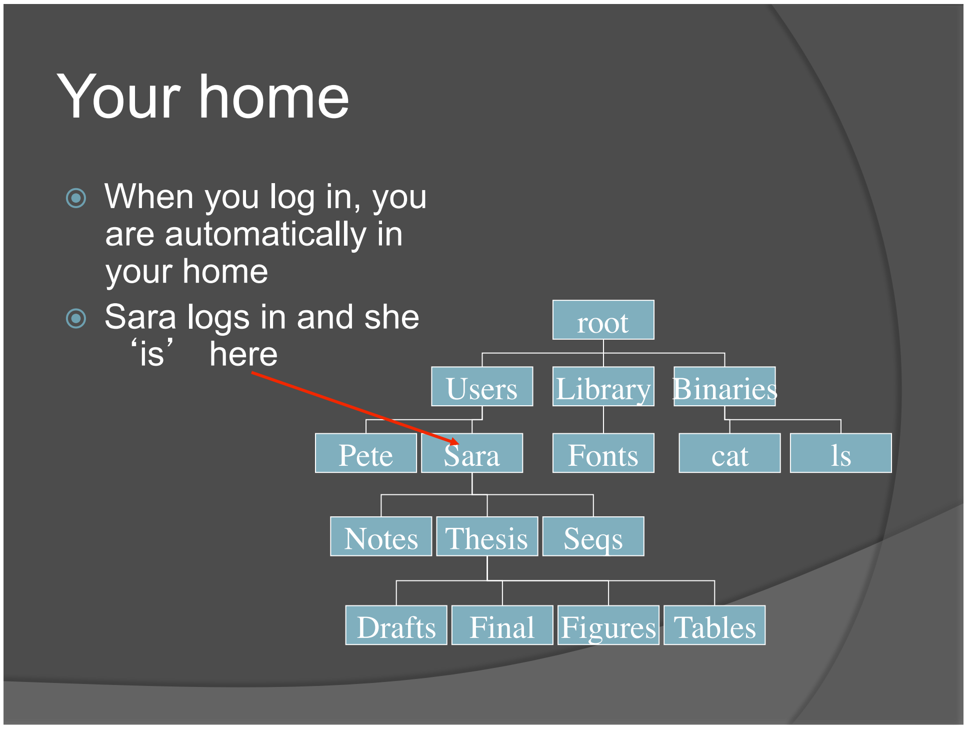
ROOT

/

The beginning of the directory system
The parent of all subdirectories

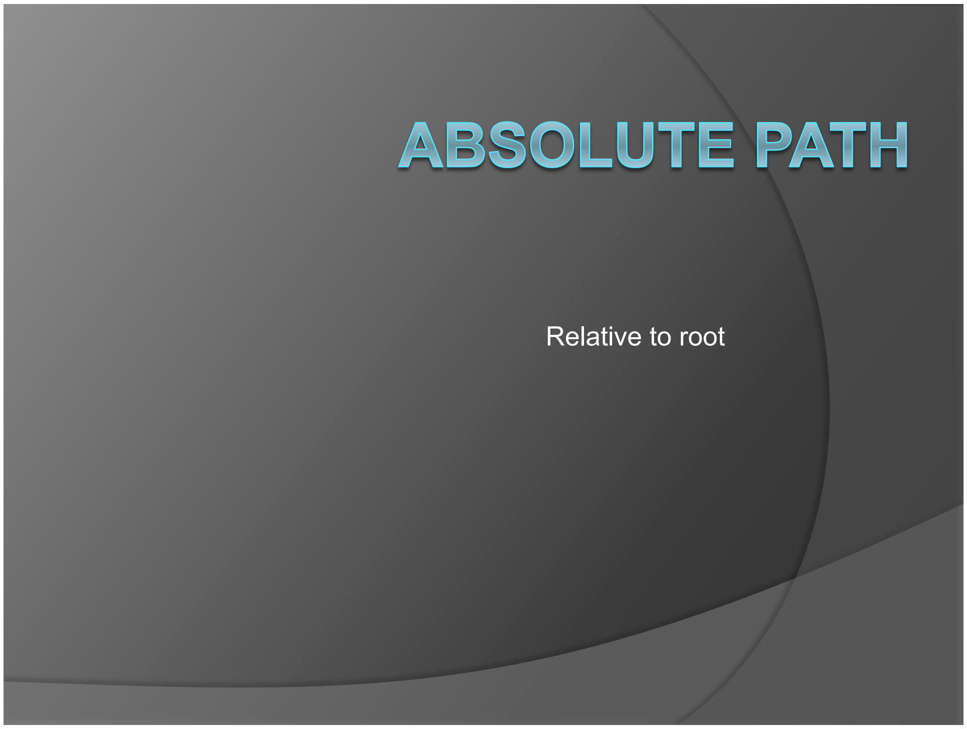
WORKING DIRECTORY

"Where you are"



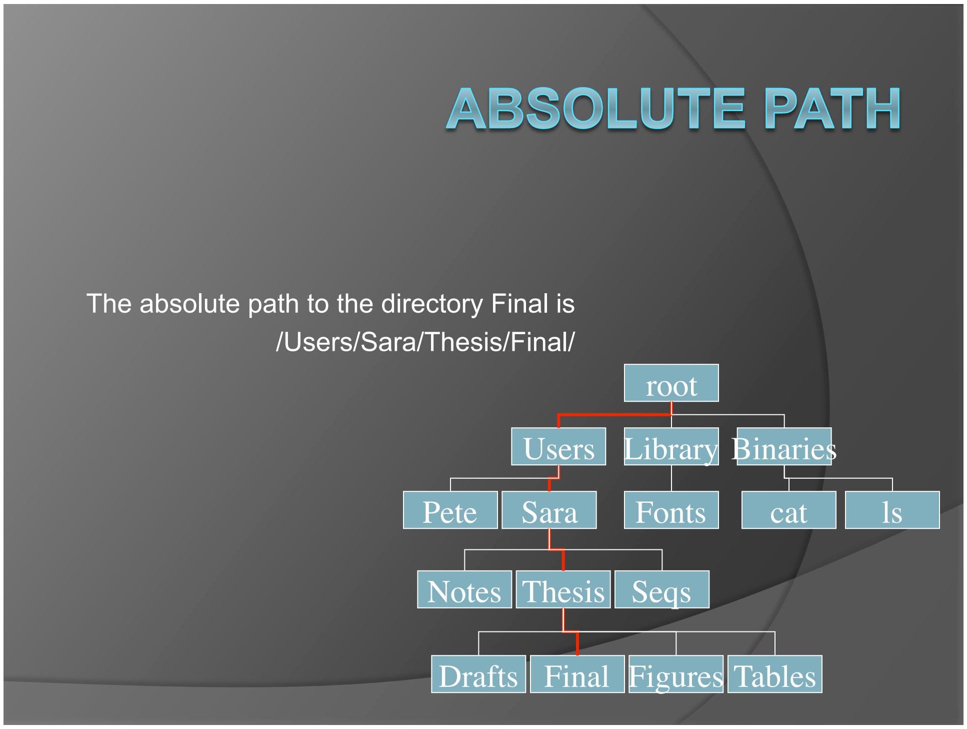
Your home

- When you log in, you are automatically in your home
- Sara logs in and she 'is' here



ABSOLUTE PATH

Relative to root



ABSOLUTE PATH

The absolute path to the directory Final is /Users/Sara/Thesis/Final/

root

Users Library Binaries

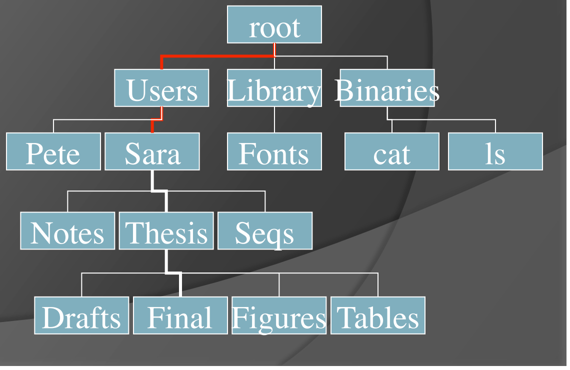
Pete Sara Fonts cat ls

Notes Thesis Seqs

Drafts Final Figures Tables

ABSOLUTE PATH

From the directory Final, the absolute path home for Sara is /Users/Sara/



root

Users Library Binaries

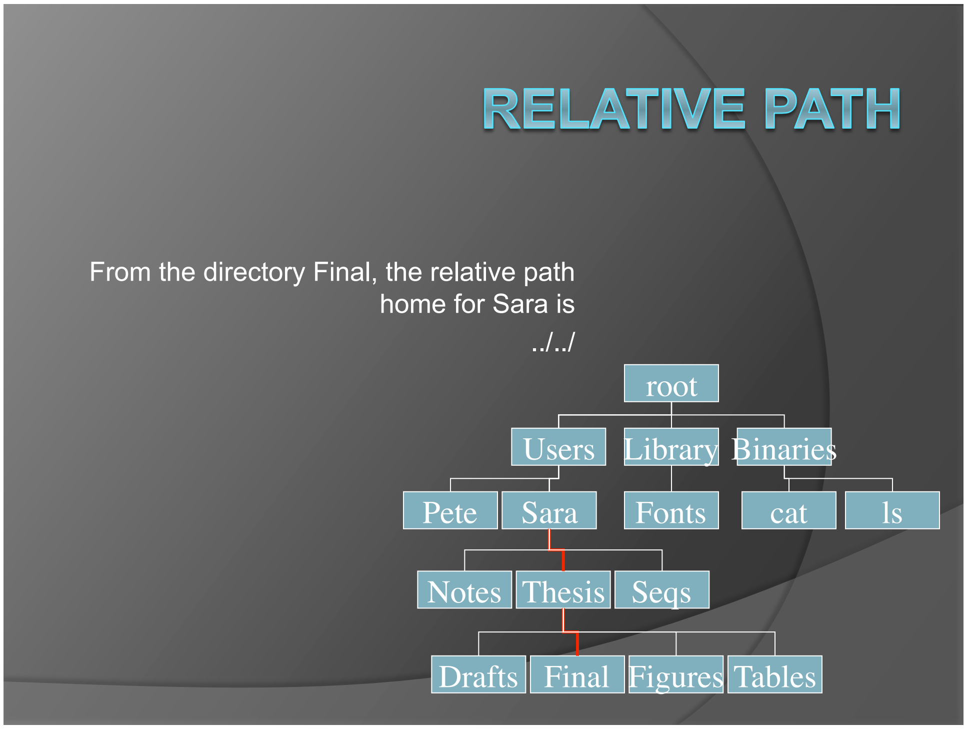
Pete Sara Fonts cat ls

Notes Thesis Seqs

Drafts Final Figures Tables

RELATIVE PATH

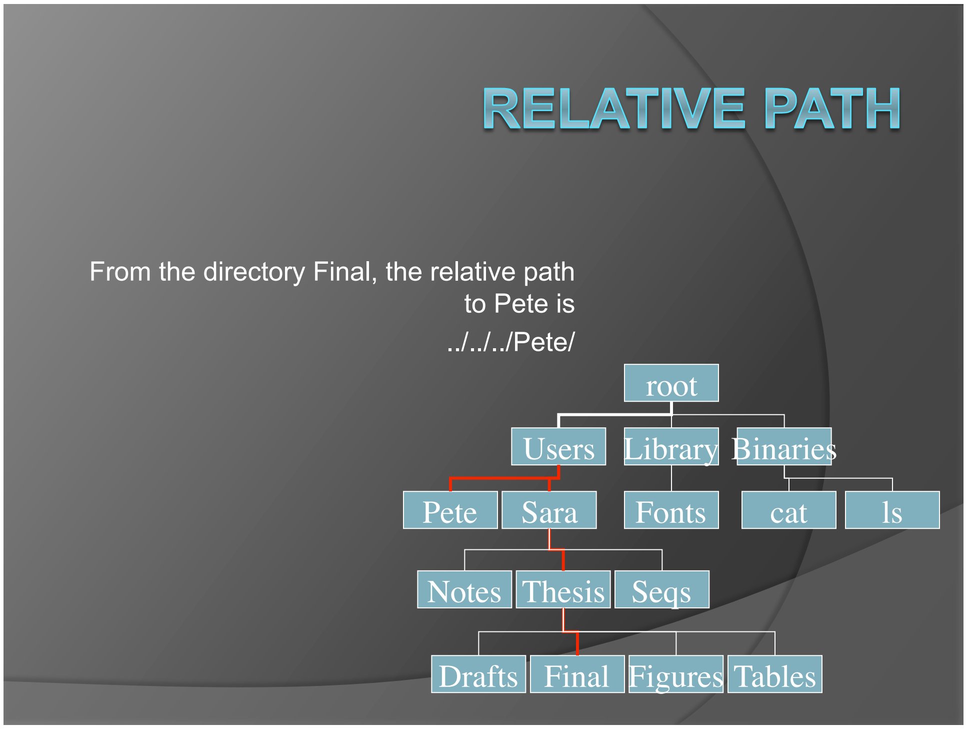
Relative to working directory



RELATIVE PATH

From the directory Final, the relative path home for Sara is

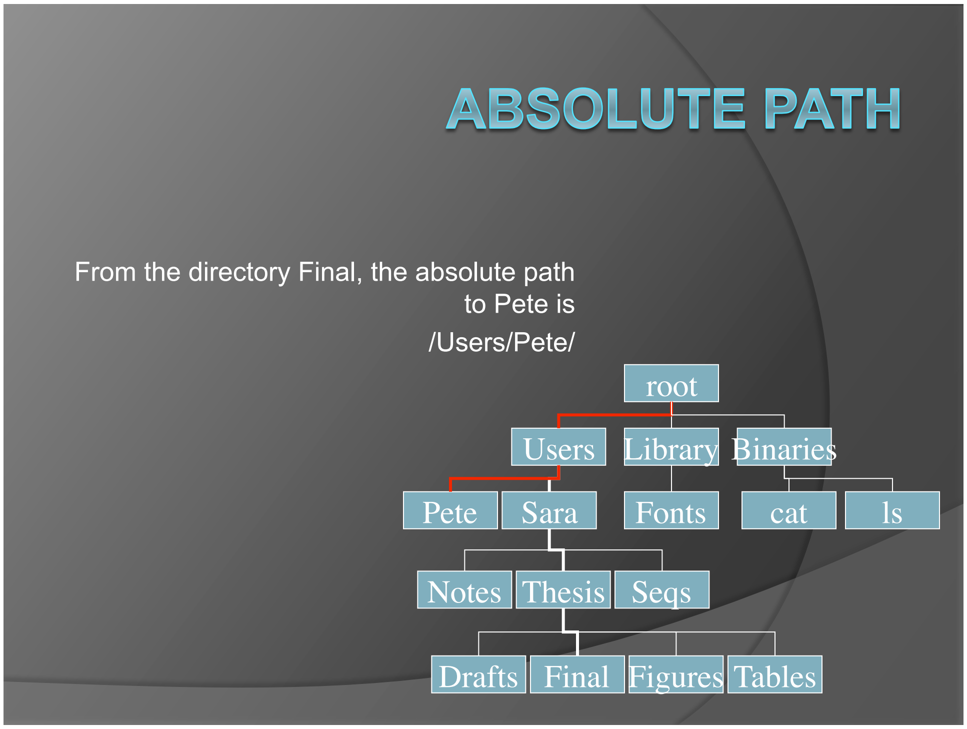
../../



RELATIVE PATH

From the directory Final, the relative path to Pete is

../../../Pete/



ABSOLUTE PATH

From the directory Final, the absolute path to Pete is

/Users/Pete/

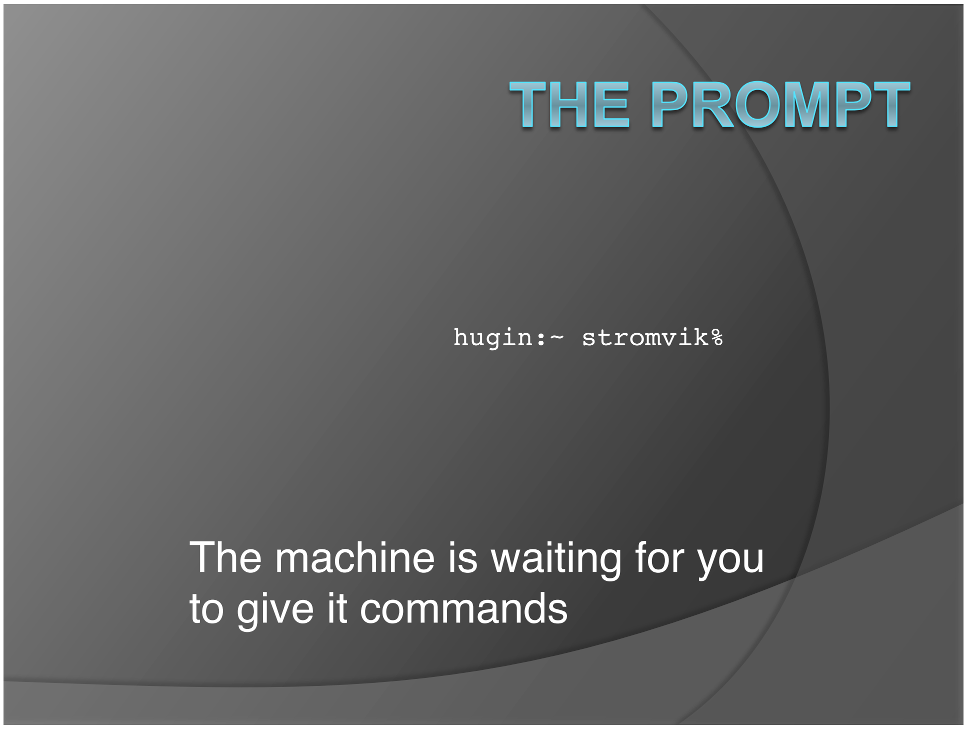
root

Users Library Binaries

Pete Sara Fonts cat ls

Notes Thesis Seqs

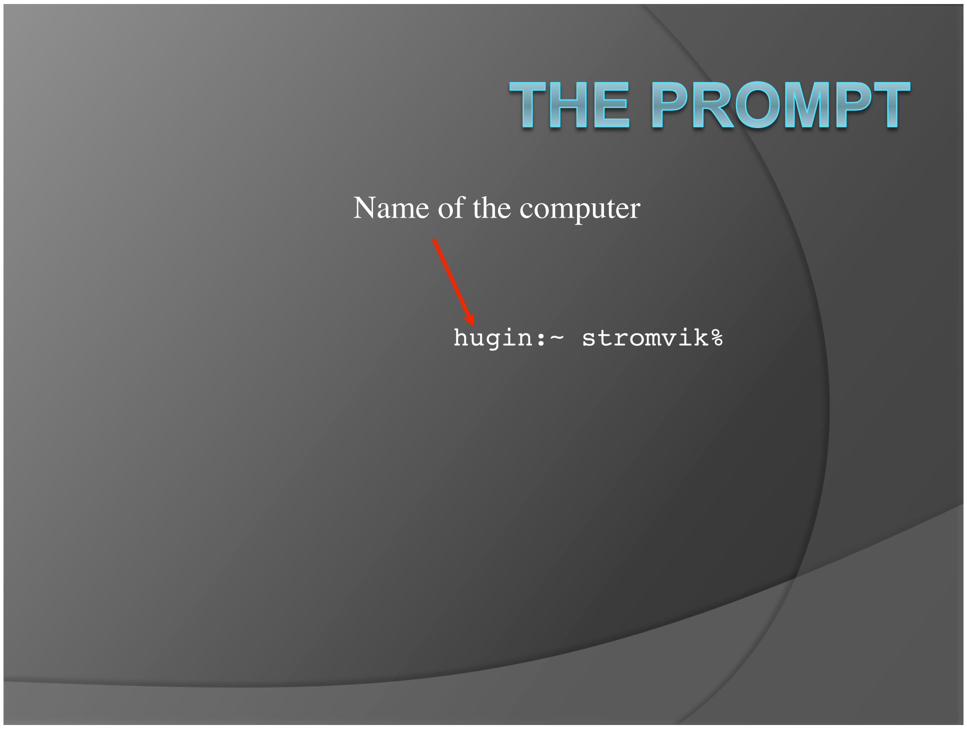
Drafts Final Figures Tables



THE PROMPT

```
hugin:~ stromvik%
```

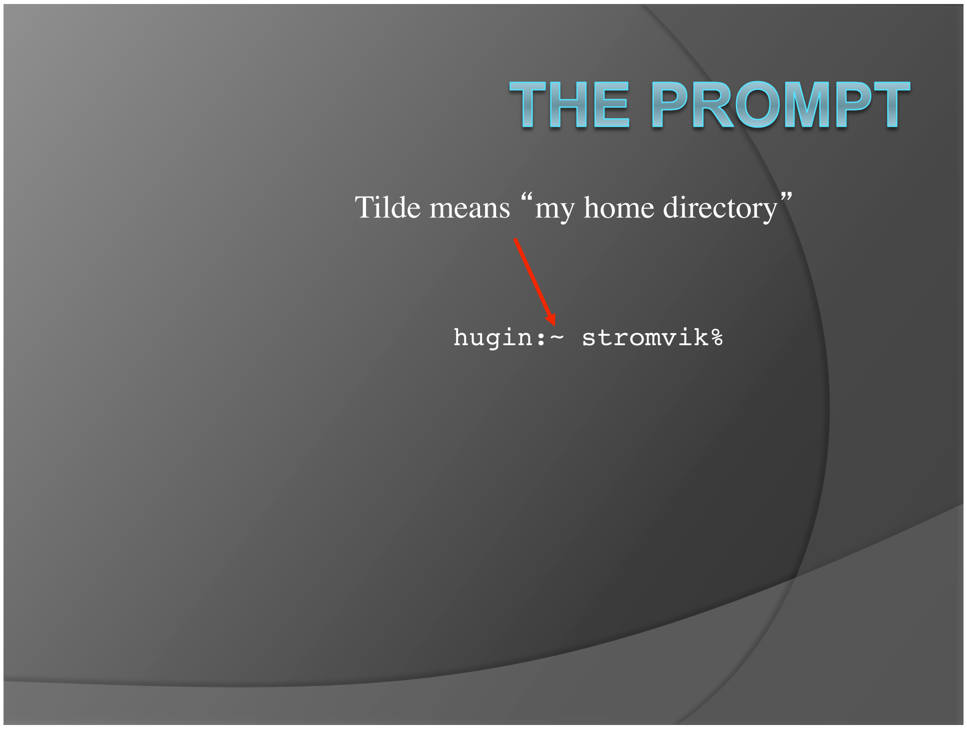
The machine is waiting for you to give it commands



THE PROMPT

Name of the computer

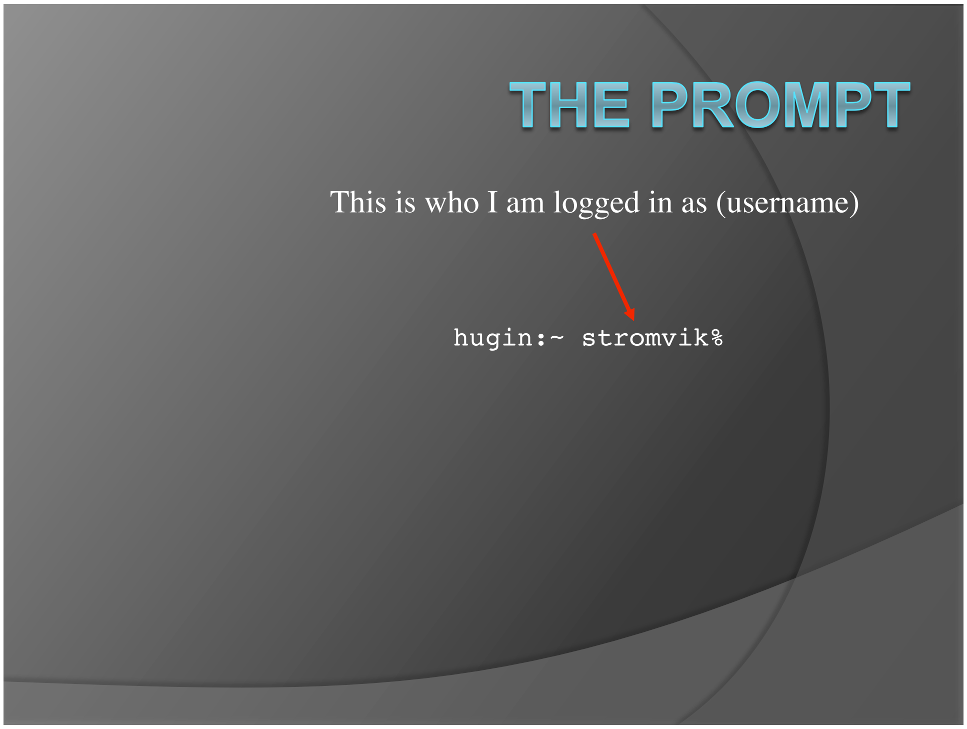
```
hugin:~ stromvik%
```



THE PROMPT

Tilde means “my home directory”

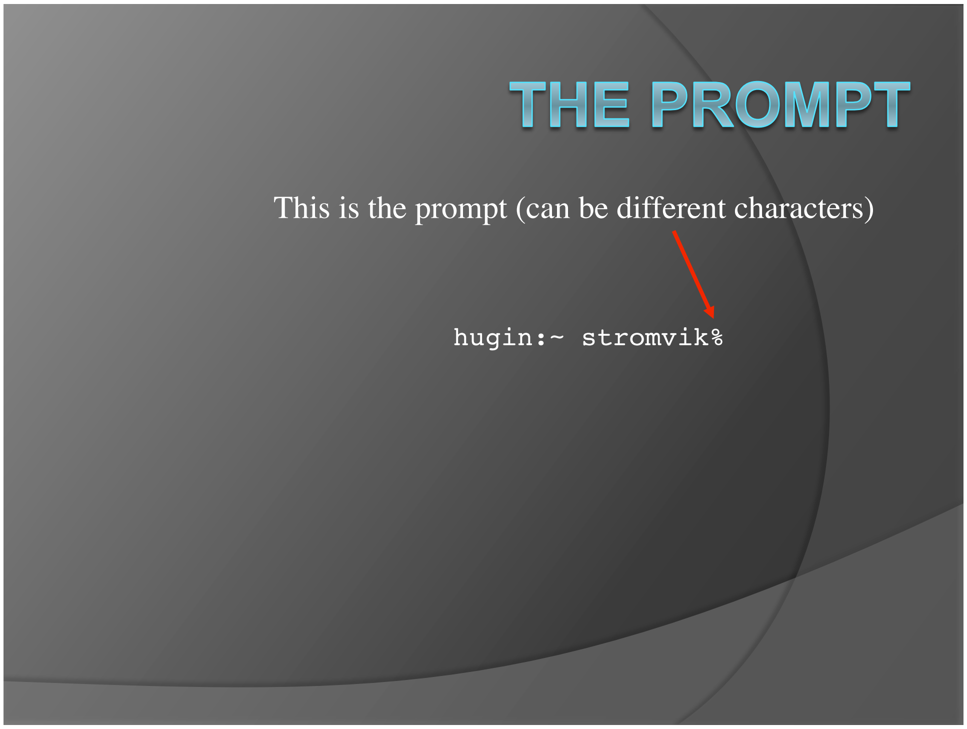
```
hugin:~ stromvik%
```



THE PROMPT

This is who I am logged in as (username)

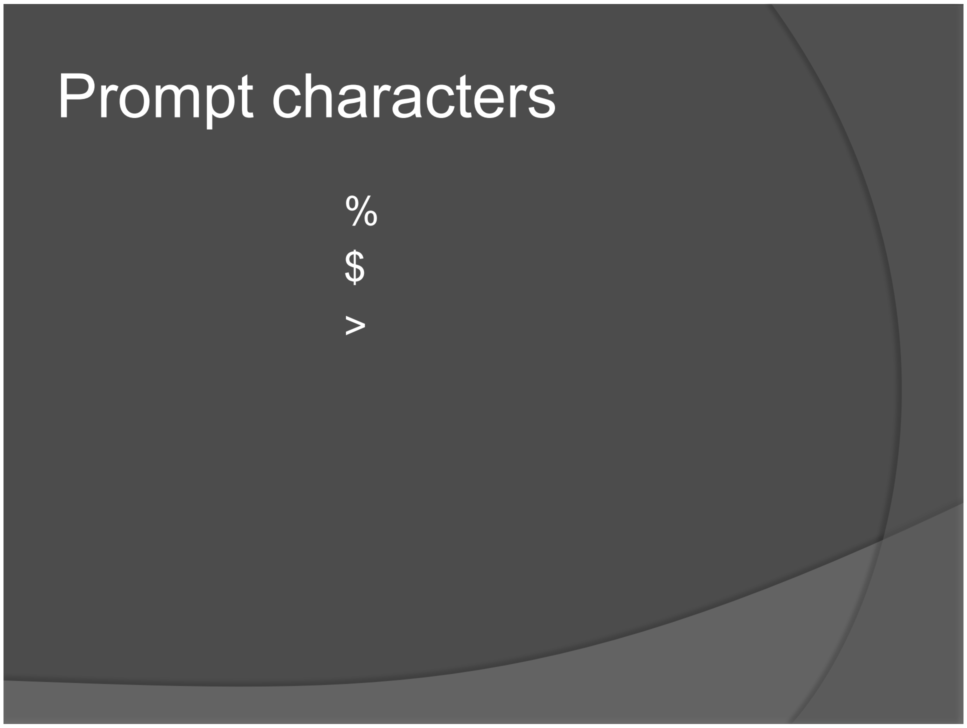
```
hugin:~ stromvik%
```



THE PROMPT

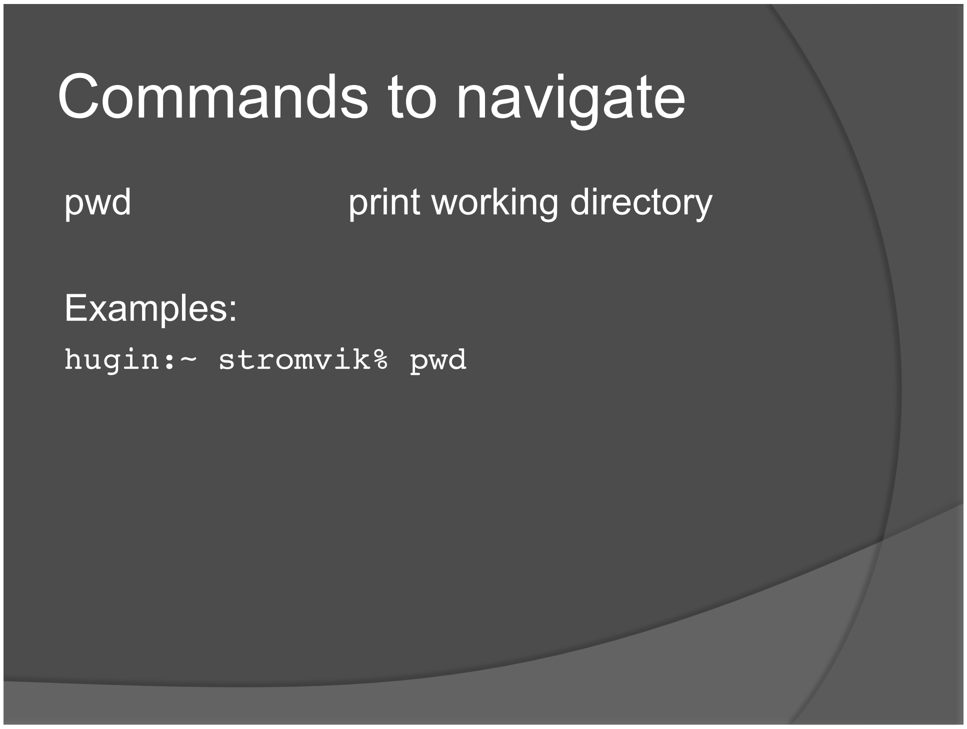
This is the prompt (can be different characters)

```
hugin:~ stromvik%
```



Prompt characters

%
$
>

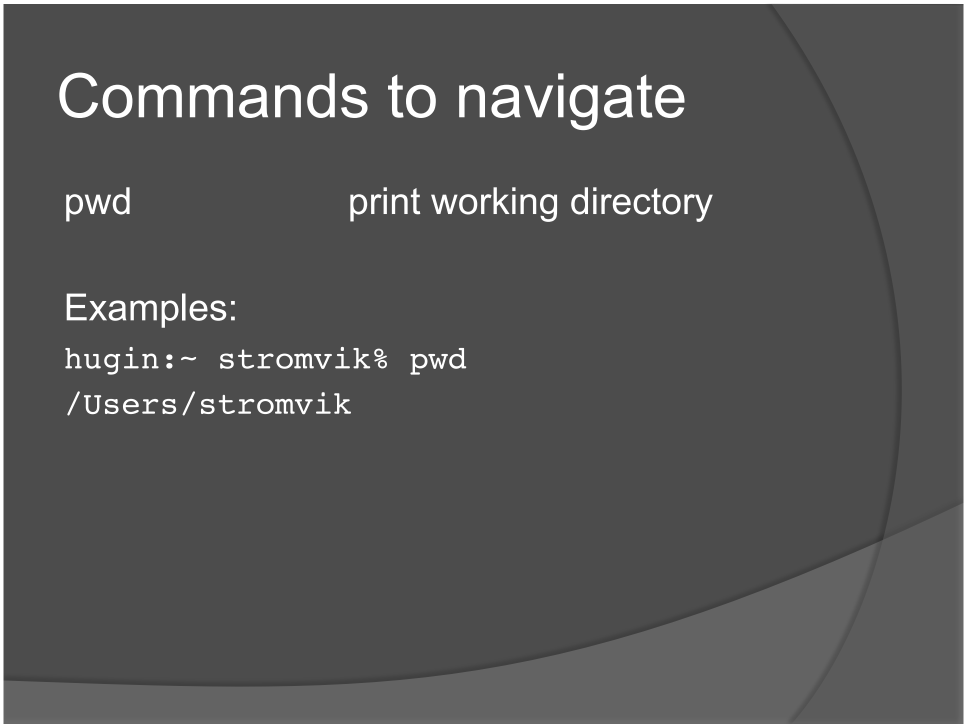


Commands to navigate

pwd print working directory

Examples:

```
hugin:~ stromvik% pwd
```

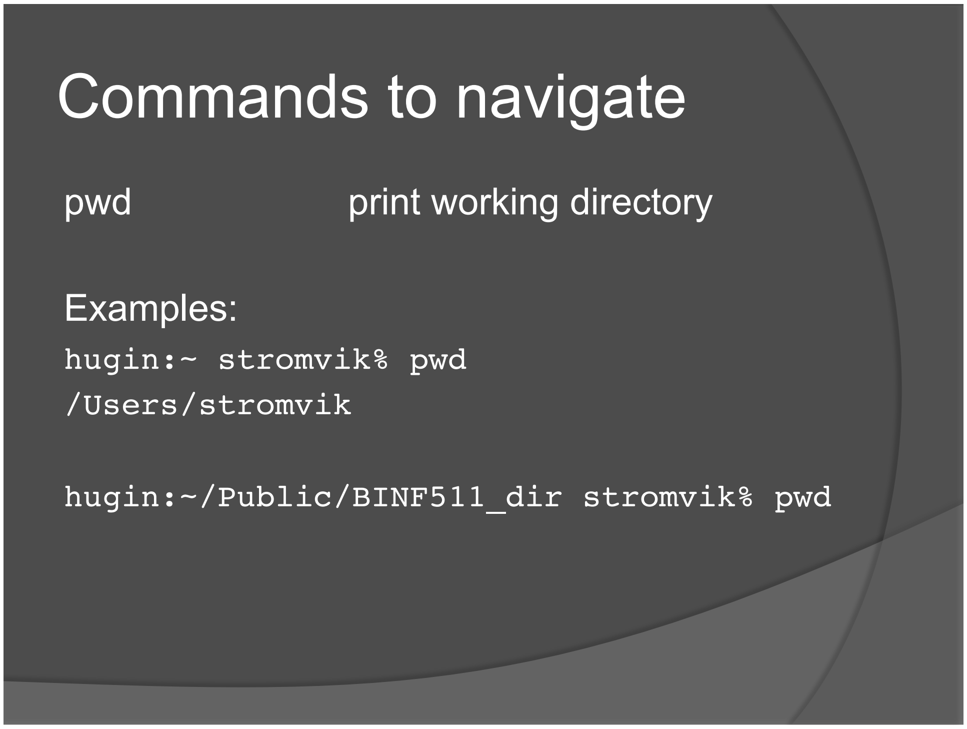


Commands to navigate

pwd print working directory

Examples:

```
hugin:~ stromvik% pwd
/Users/stromvik
```

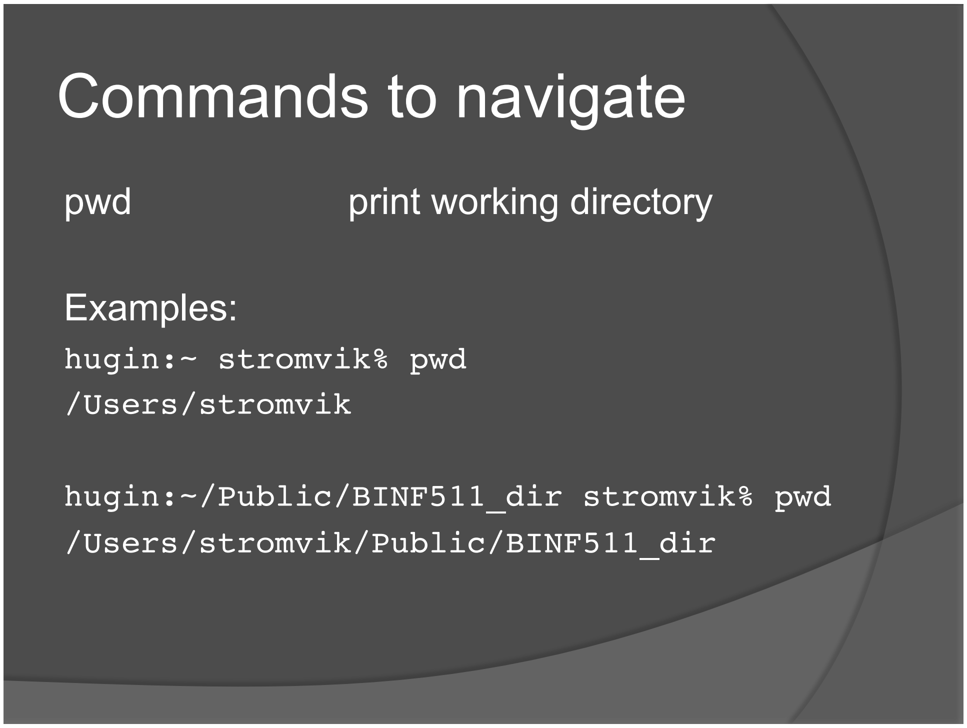


Commands to navigate

pwd print working directory

Examples:

```
hugin:~ stromvik% pwd
/Users/stromvik

hugin:~/Public/BINF511_dir stromvik% pwd
```

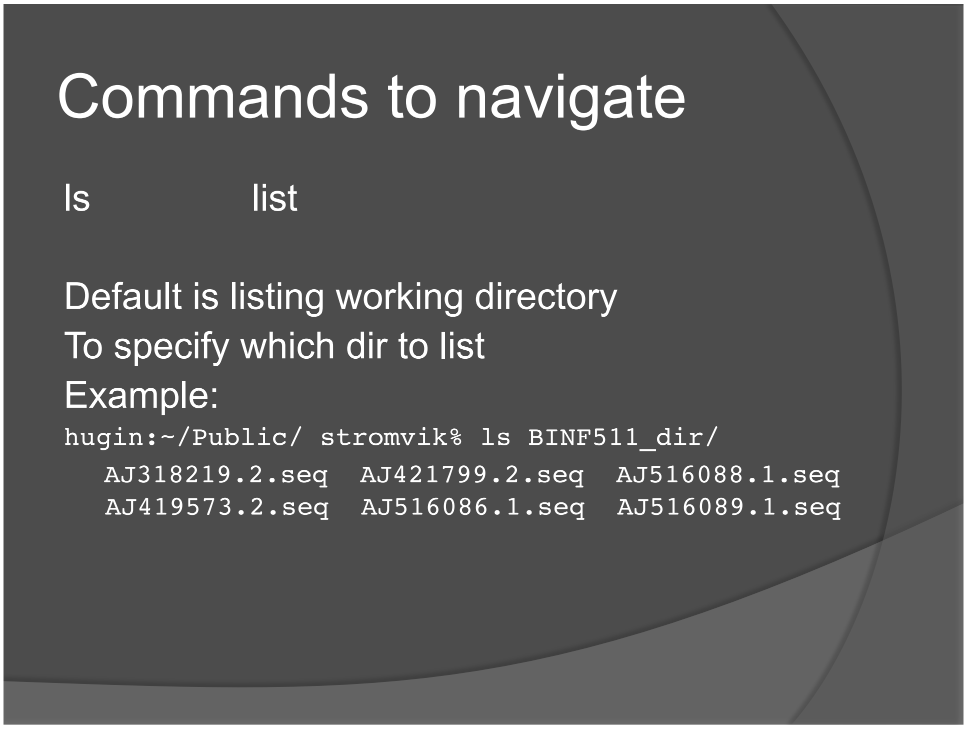


Commands to navigate

pwd print working directory

Examples:

```
hugin:~ stromvik% pwd
/Users/stromvik

hugin:~/Public/BINF511_dir stromvik% pwd
/Users/stromvik/Public/BINF511_dir
```



Commands to navigate

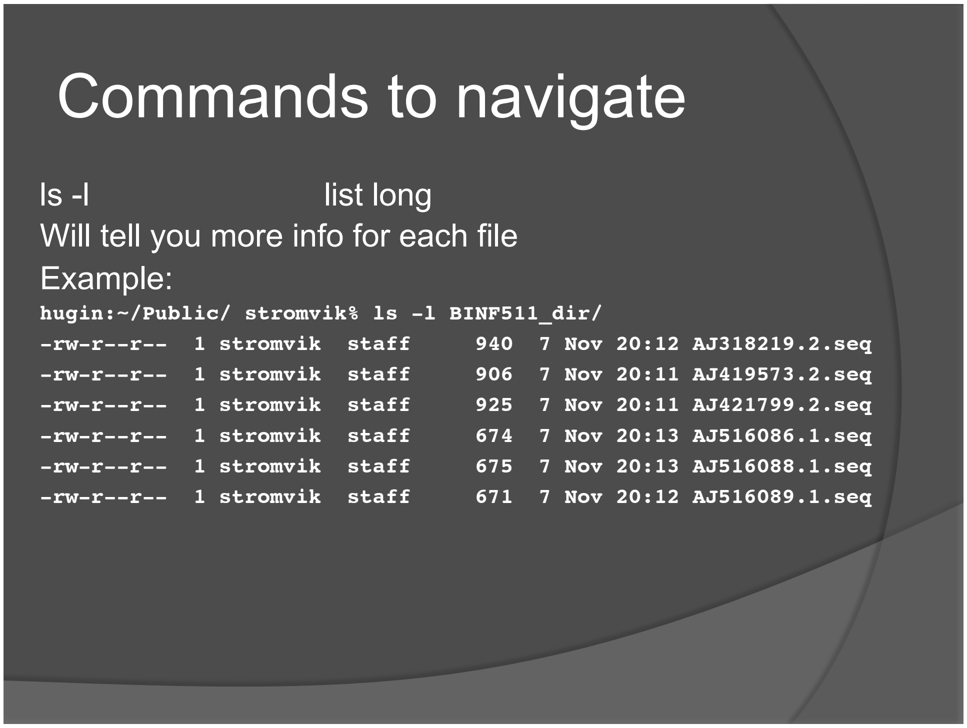
ls list

Default is listing working directory

To specify which dir to list

Example:

```
hugin:~/Public/ stromvik% ls BINF511_dir/
   AJ318219.2.seq  AJ421799.2.seq  AJ516088.1.seq
   AJ419573.2.seq  AJ516086.1.seq  AJ516089.1.seq
```



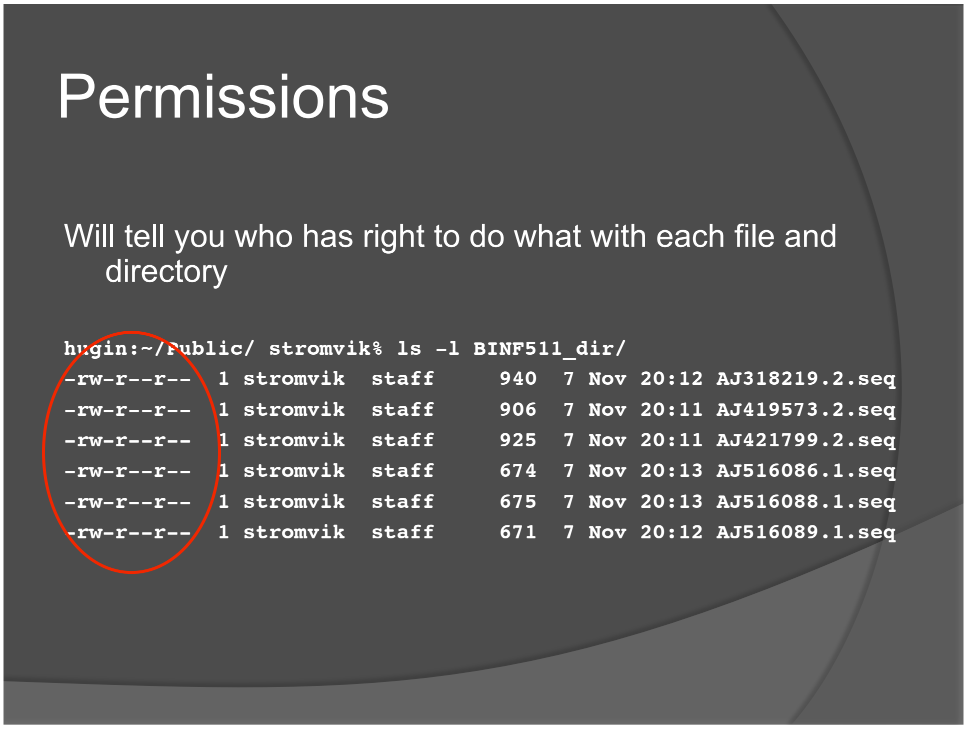
Commands to navigate

ls -l list long

Will tell you more info for each file

Example:

```
hugin:~/Public/ stromvik% ls -l BINF511_dir/
-rw-r--r--  1 stromvik  staff    940  7 Nov 20:12 AJ318219.2.seq
-rw-r--r--  1 stromvik  staff    906  7 Nov 20:11 AJ419573.2.seq
-rw-r--r--  1 stromvik  staff    925  7 Nov 20:11 AJ421799.2.seq
-rw-r--r--  1 stromvik  staff    674  7 Nov 20:13 AJ516086.1.seq
-rw-r--r--  1 stromvik  staff    675  7 Nov 20:13 AJ516088.1.seq
-rw-r--r--  1 stromvik  staff    671  7 Nov 20:12 AJ516089.1.seq
```

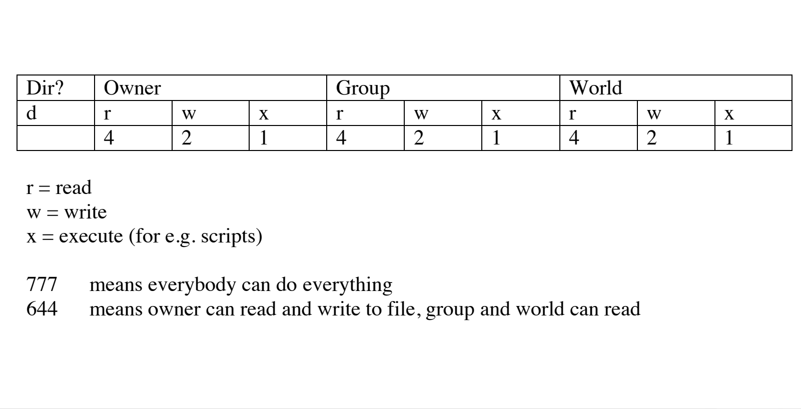


Permissions

Will tell you who has right to do what with each file and directory

```
hugin:~/Public/ stromvik% ls -l BINF511_dir/
-rw-r--r--  1 stromvik  staff    940  7 Nov 20:12 AJ318219.2.seq
-rw-r--r--  1 stromvik  staff    906  7 Nov 20:11 AJ419573.2.seq
-rw-r--r--  1 stromvik  staff    925  7 Nov 20:11 AJ421799.2.seq
-rw-r--r--  1 stromvik  staff    674  7 Nov 20:13 AJ516086.1.seq
-rw-r--r--  1 stromvik  staff    675  7 Nov 20:13 AJ516088.1.seq
-rw-r--r--  1 stromvik  staff    671  7 Nov 20:12 AJ516089.1.seq
```

Permissions

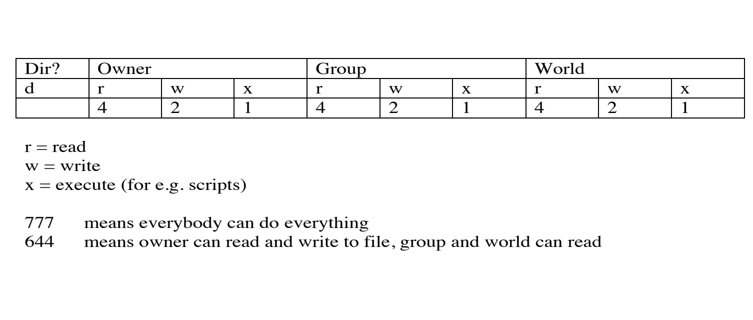


Dir?	Owner			Group			World		
d	r	w	x	r	w	x	r	w	x
	4	2	1	4	2	1	4	2	1

r = read
w = write
x = execute (for e.g. scripts)

777 means everybody can do everything
644 means owner can read and write to file, group and world can read

Permissions (mode)



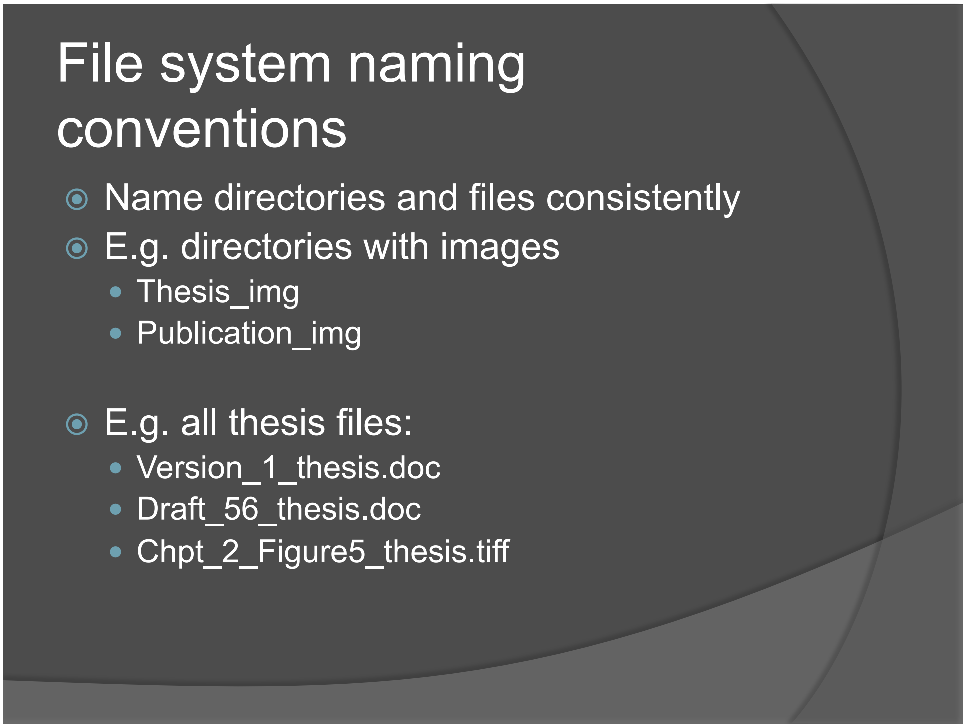
Dir?	Owner			Group			World		
d	r	w	x	r	w	x	r	w	x
	4	2	1	4	2	1	4	2	1

r = read
w = write
x = execute (for e.g. scripts)

777 means everybody can do everything
644 means owner can read and write to file, group and world can read

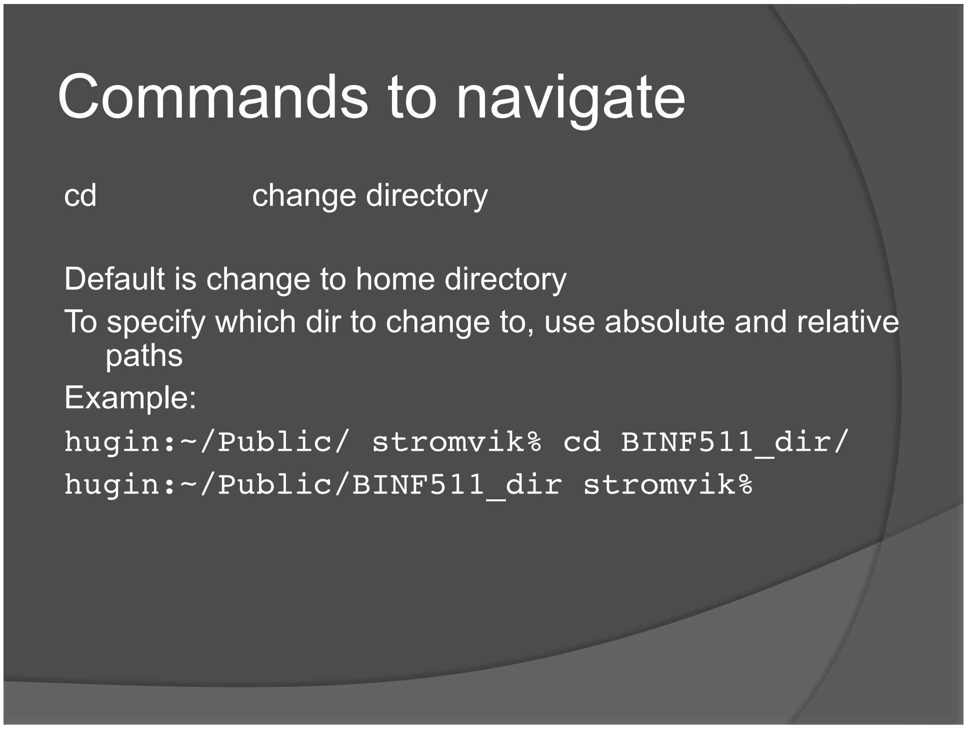
chmod 777 *mytestfile.txt*

will change the permissions on the file mytestfile.txt



File system naming conventions

- Name directories and files consistently
- E.g. directories with images
  - Thesis_img
  - Publication_img
- E.g. all thesis files:
  - Version_1_thesis.doc
  - Draft_56_thesis.doc
  - Chpt_2_Figure5_thesis.tiff



Commands to navigate

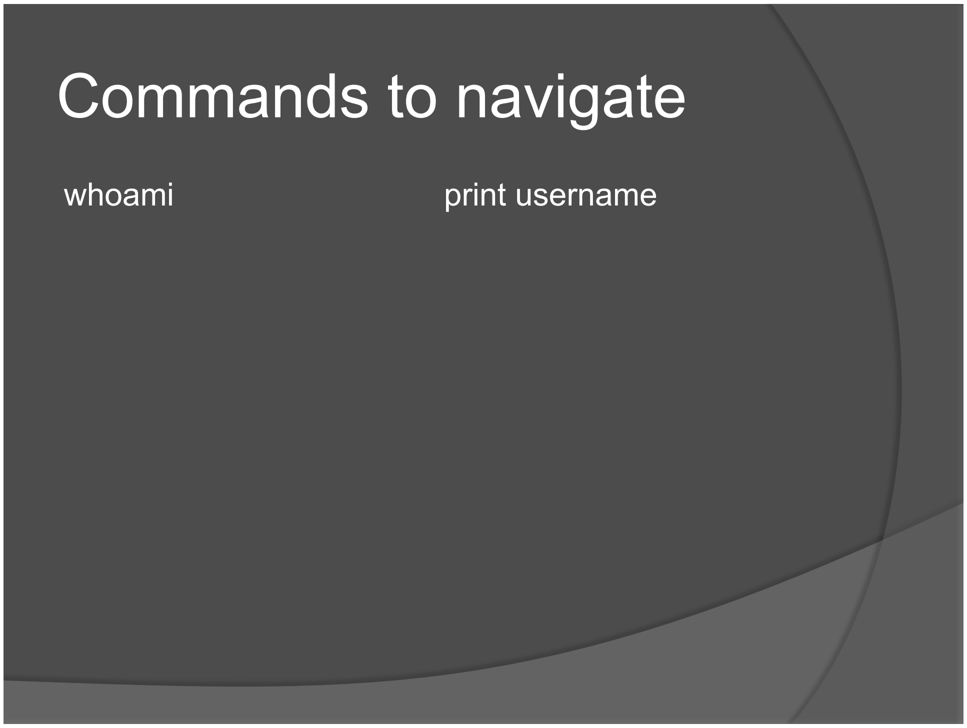
cd change directory

Default is change to home directory

To specify which dir to change to, use absolute and relative paths

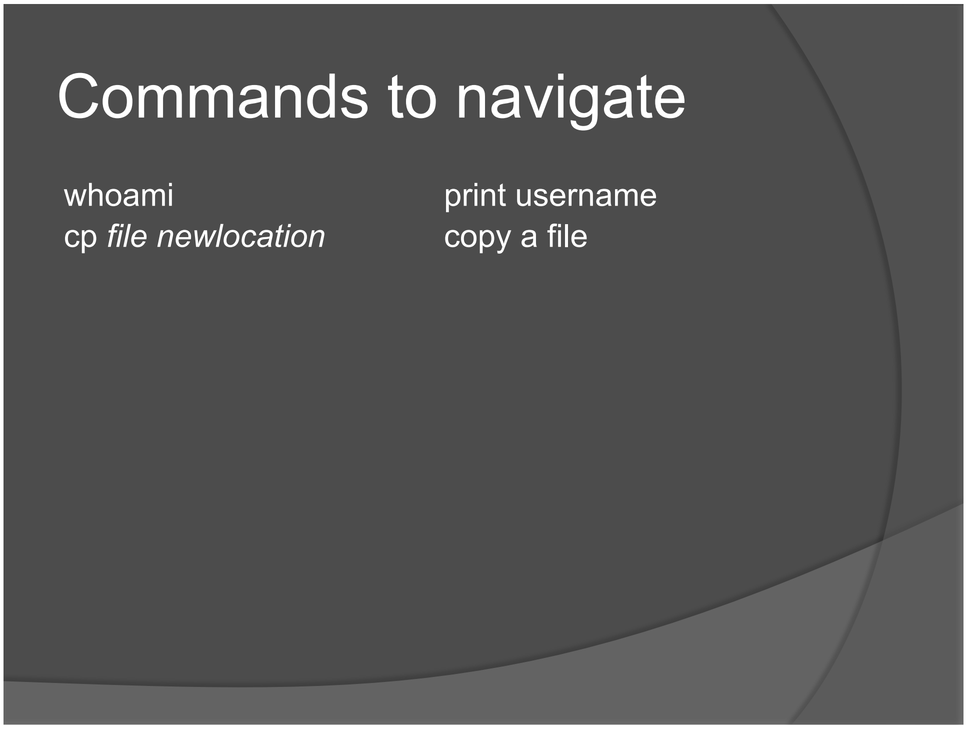
Example:

```
hugin:~/Public/ stromvik% cd BINF511_dir/
hugin:~/Public/BINF511_dir stromvik%
```



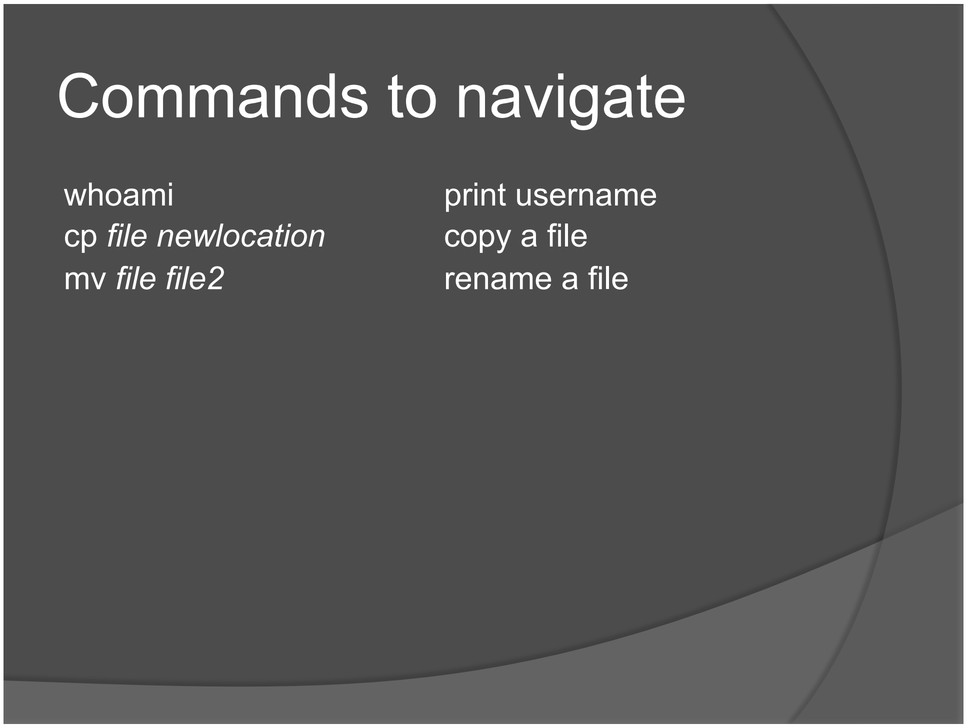
Commands to navigate

whoami	print username



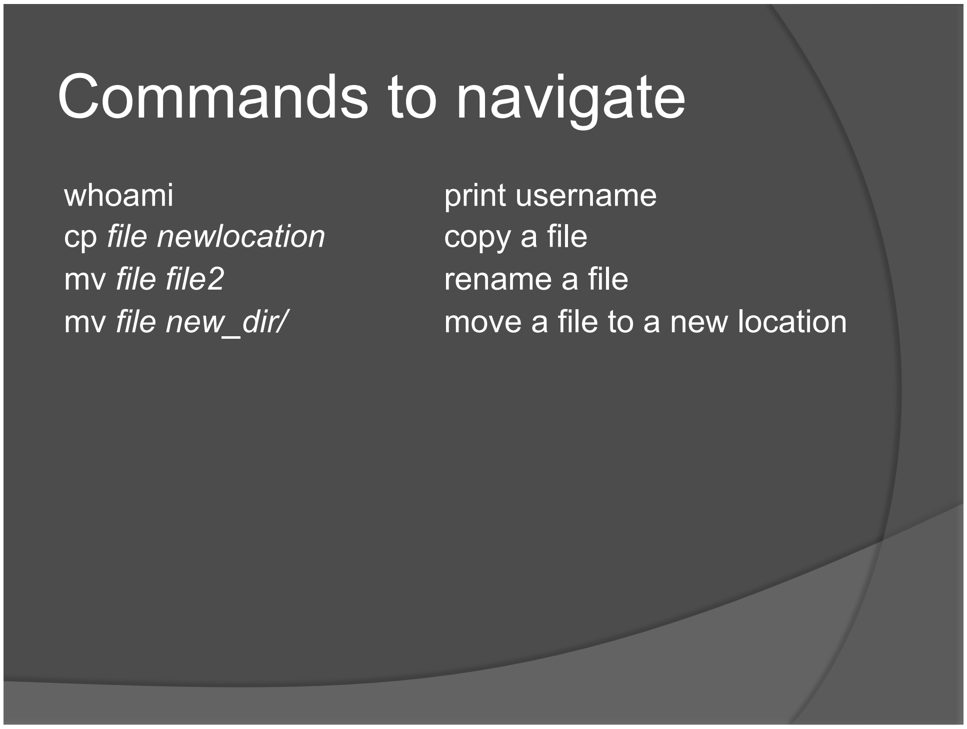
Commands to navigate

whoami	print username
cp *file newlocation*	copy a file



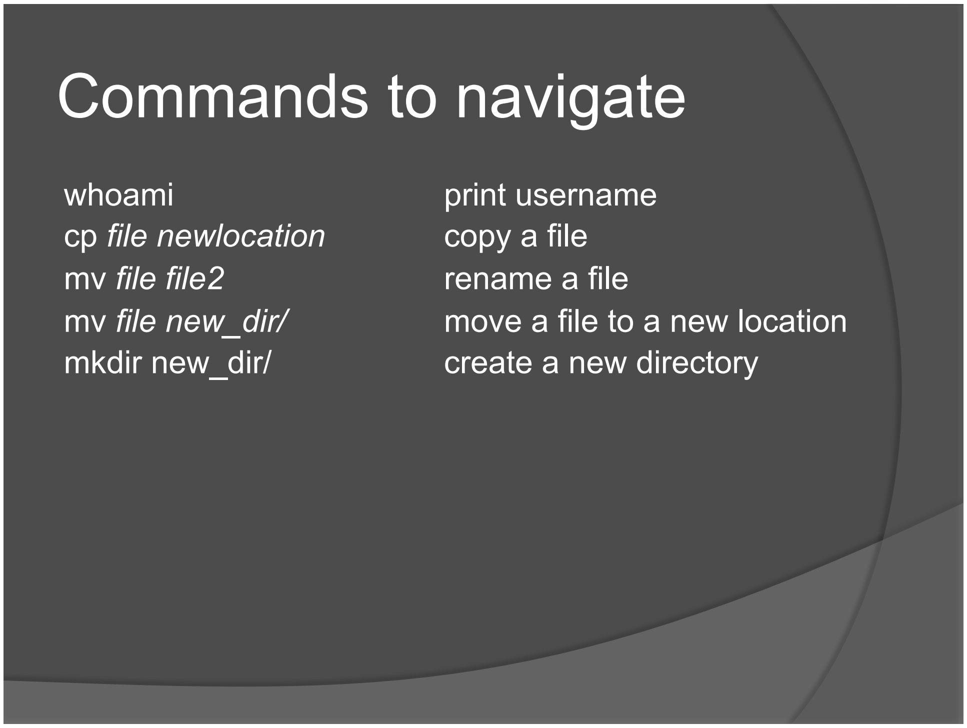
Commands to navigate

whoami	print username
cp *file newlocation*	copy a file
mv *file file2*	rename a file



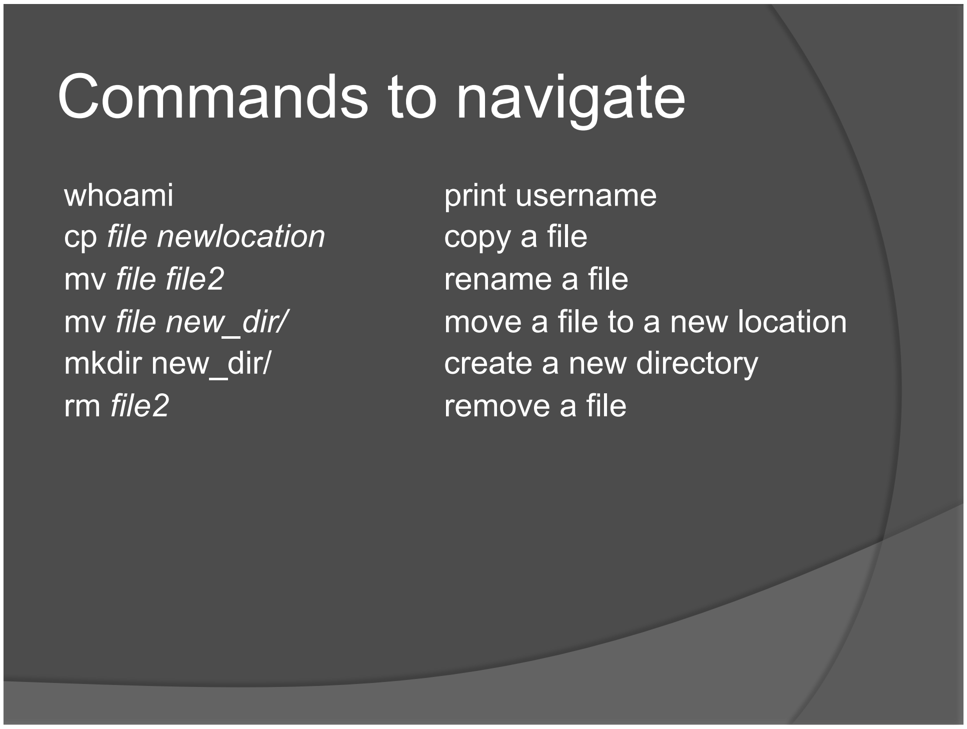
Commands to navigate

whoami	print username
cp *file newlocation*	copy a file
mv *file file2*	rename a file
mv *file new_dir/*	move a file to a new location



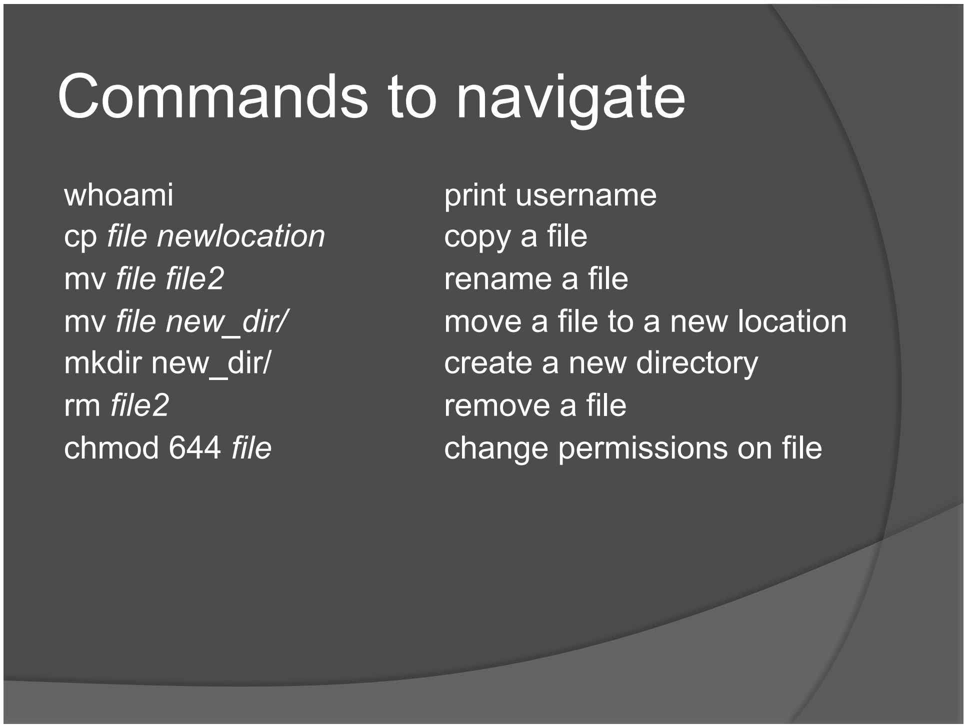
Commands to navigate

whoami	print username
cp *file newlocation*	copy a file
mv *file file2*	rename a file
mv *file new_dir/*	move a file to a new location
mkdir new_dir/	create a new directory



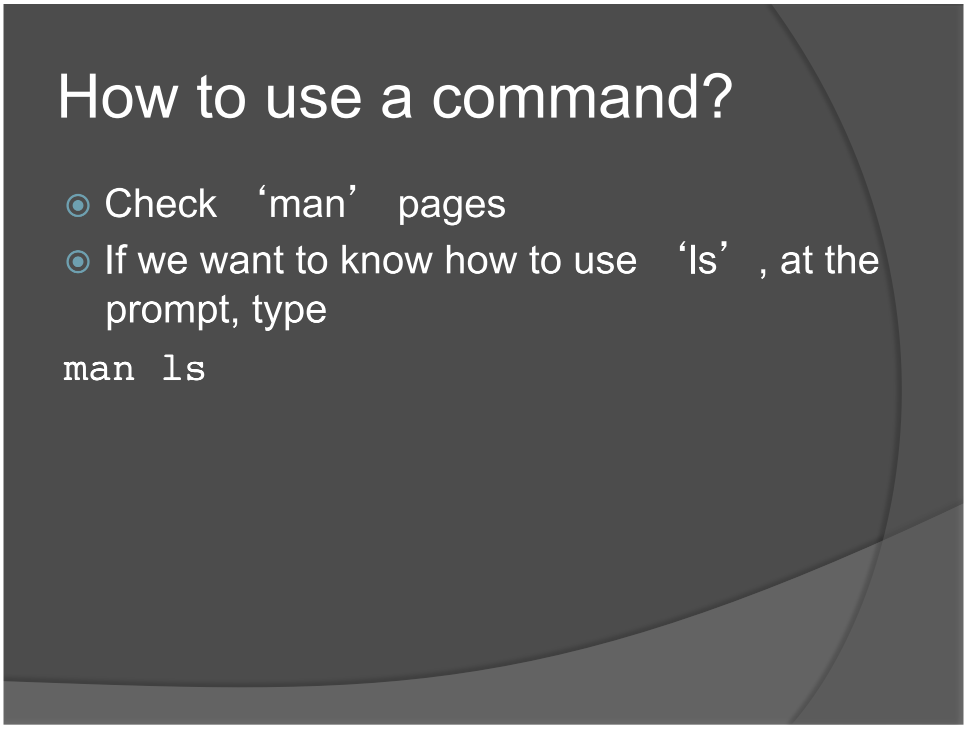
Commands to navigate

whoami	print username
cp *file newlocation*	copy a file
mv *file file2*	rename a file
mv *file new_dir/*	move a file to a new location
mkdir new_dir/	create a new directory
rm *file2*	remove a file



Commands to navigate

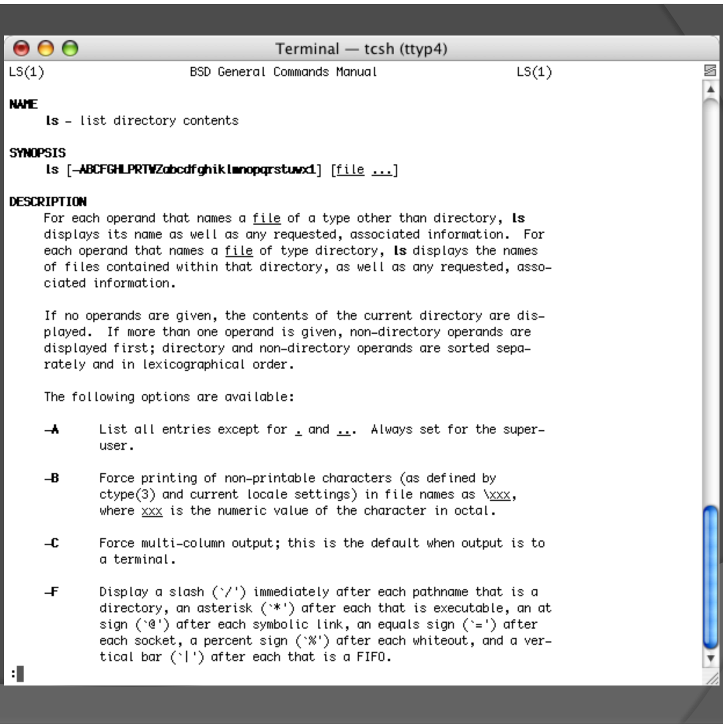
whoami	print username
cp *file newlocation*	copy a file
mv *file file2*	rename a file
mv *file new_dir/*	move a file to a new location
mkdir new_dir/	create a new directory
rm *file2*	remove a file
chmod 644 *file*	change permissions on file



How to use a command?

- Check ‘man’ pages
- If we want to know how to use ‘ls’, at the prompt, type

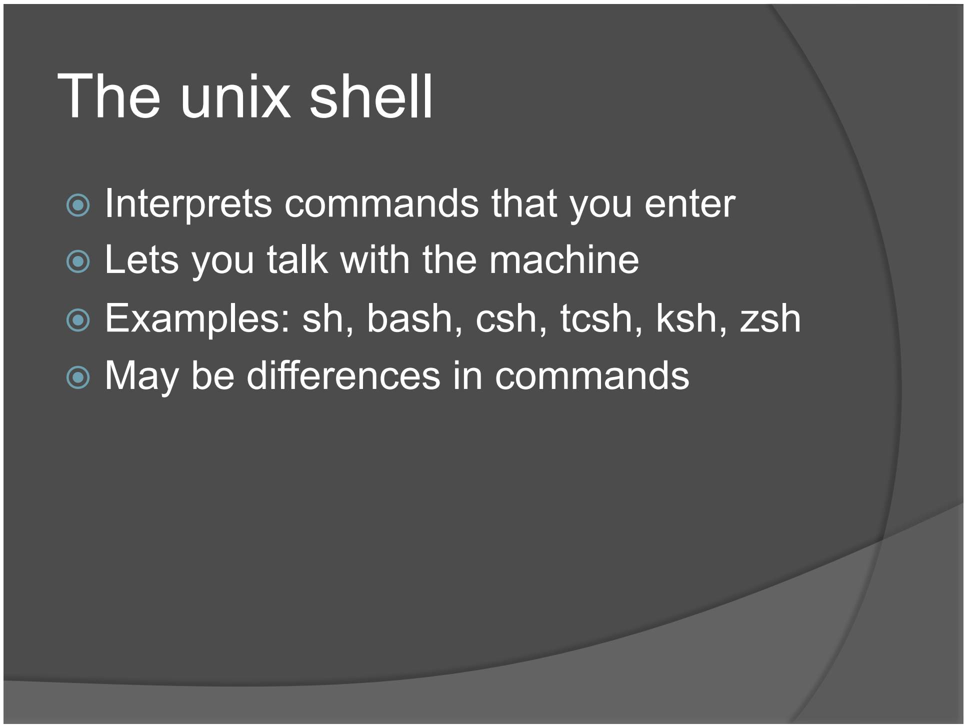
```
man ls
```



```
LS(1)                   BSD General Commands Manual                  LS(1)

NAME
     ls - list directory contents

SYNOPSIS
     ls [-ABCFGHLPRTWZabcdfghiklmnopqrstuwx1] [file ...]

DESCRIPTION
     For each operand that names a file of a type other than directory, ls
     displays its name as well as any requested, associated information.  For
     each operand that names a file of type directory, ls displays the names
     of files contained within that directory, as well as any requested, asso-
     ciated information.

     If no operands are given, the contents of the current directory are dis-
     played.  If more than one operand is given, non-directory operands are
     displayed first; directory and non-directory operands are sorted sepa-
     rately and in lexicographical order.

     The following options are available:

     -A      List all entries except for . and ...  Always set for the super-
             user.

     -B      Force printing of non-printable characters (as defined by
             ctype(3) and current locale settings) in file names as \xxx,
             where xxx is the numeric value of the character in octal.

     -C      Force multi-column output; this is the default when output is to
             a terminal.

     -F      Display a slash (`/') immediately after each pathname that is a
             directory, an asterisk (`*') after each that is executable, an at
             sign (`@') after each symbolic link, an equals sign (`=') after
             each socket, a percent sign (`%') after each whiteout, and a ver-
             tical bar (`|') after each that is a FIFO.
:
```

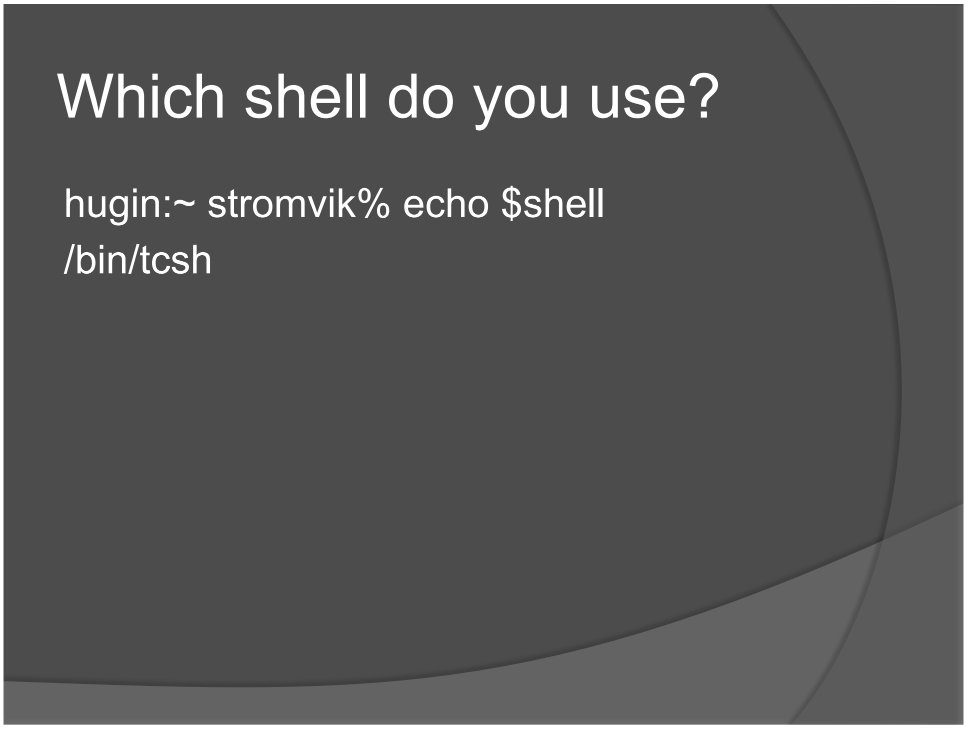
SHELLS

http://www.seashellworld.com/



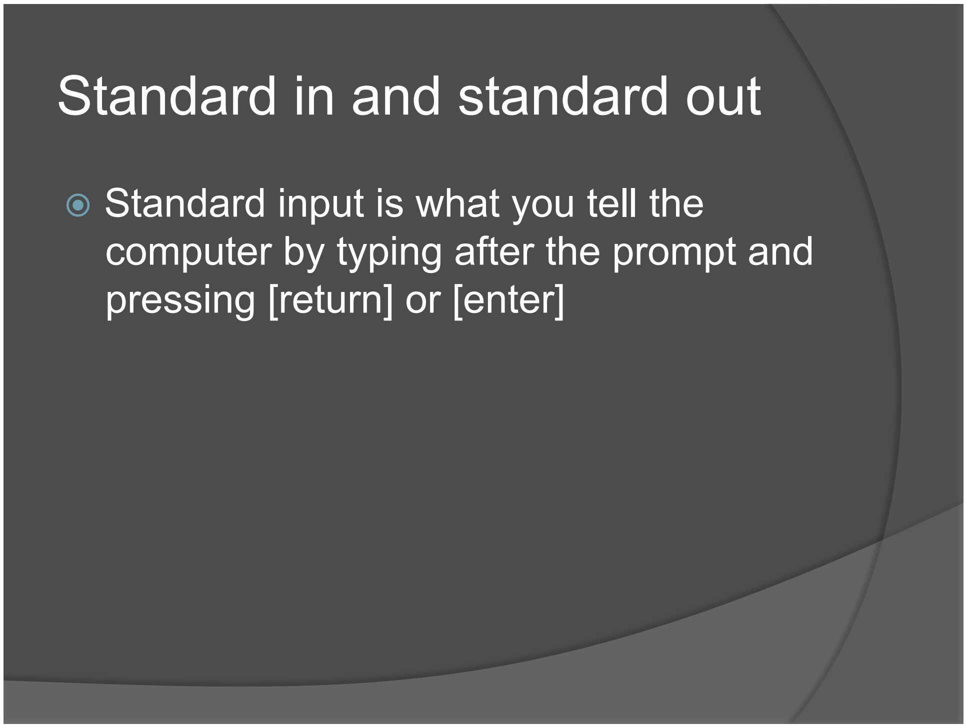
The unix shell

- Interprets commands that you enter
- Lets you talk with the machine
- Examples: sh, bash, csh, tcsh, ksh, zsh
- May be differences in commands



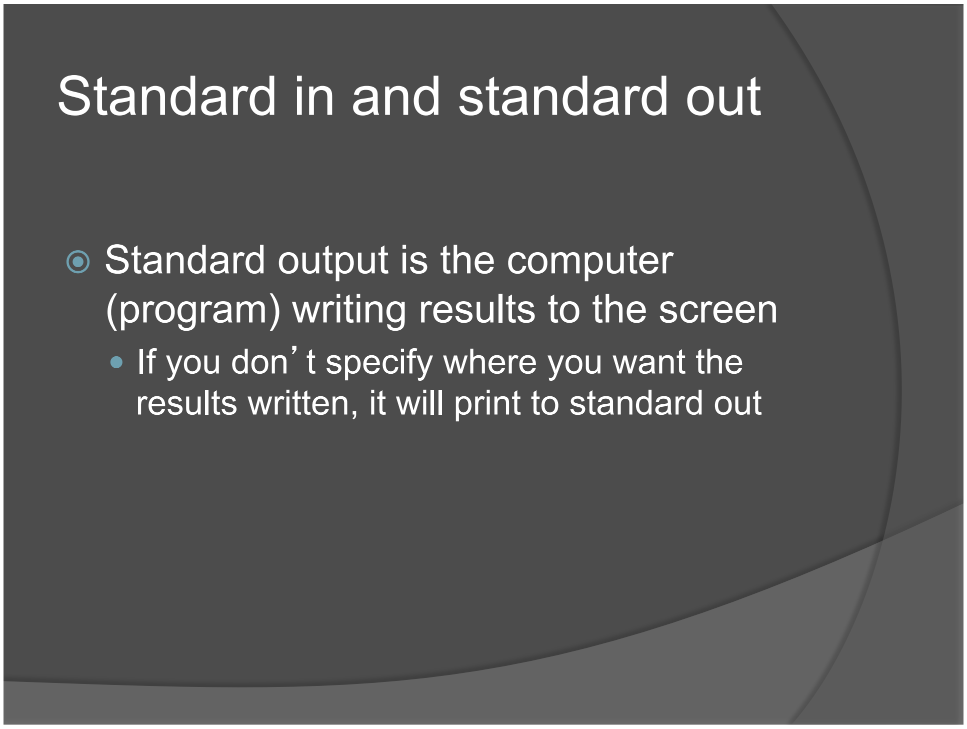
Which shell do you use?

```
hugin:~ stromvik% echo $shell
/bin/tcsh
```



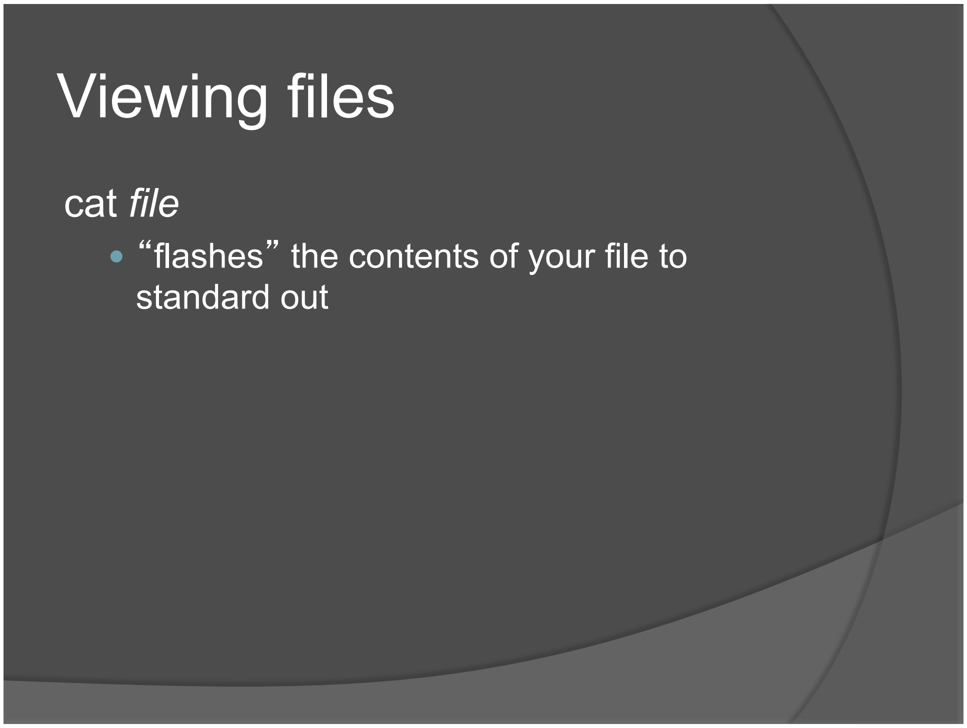
Standard in and standard out

- Standard input is what you tell the computer by typing after the prompt and pressing [return] or [enter]



Standard in and standard out

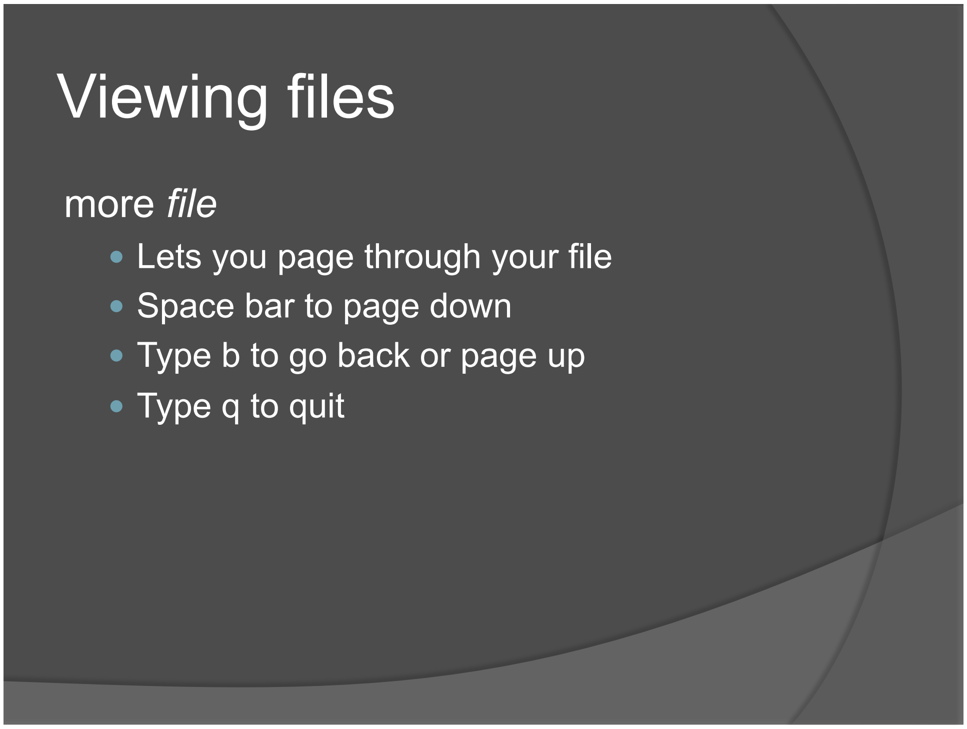
- Standard output is the computer (program) writing results to the screen
  - If you don’t specify where you want the results written, it will print to standard out



Viewing files

cat *file*

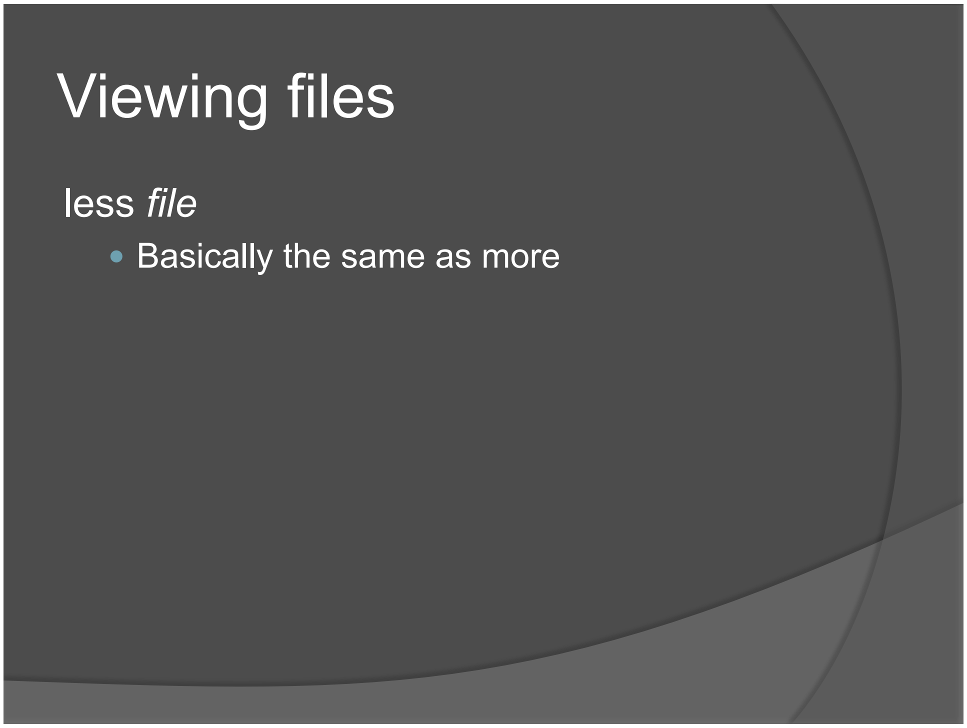
- “flashes” the contents of your file to standard out



Viewing files

more *file*

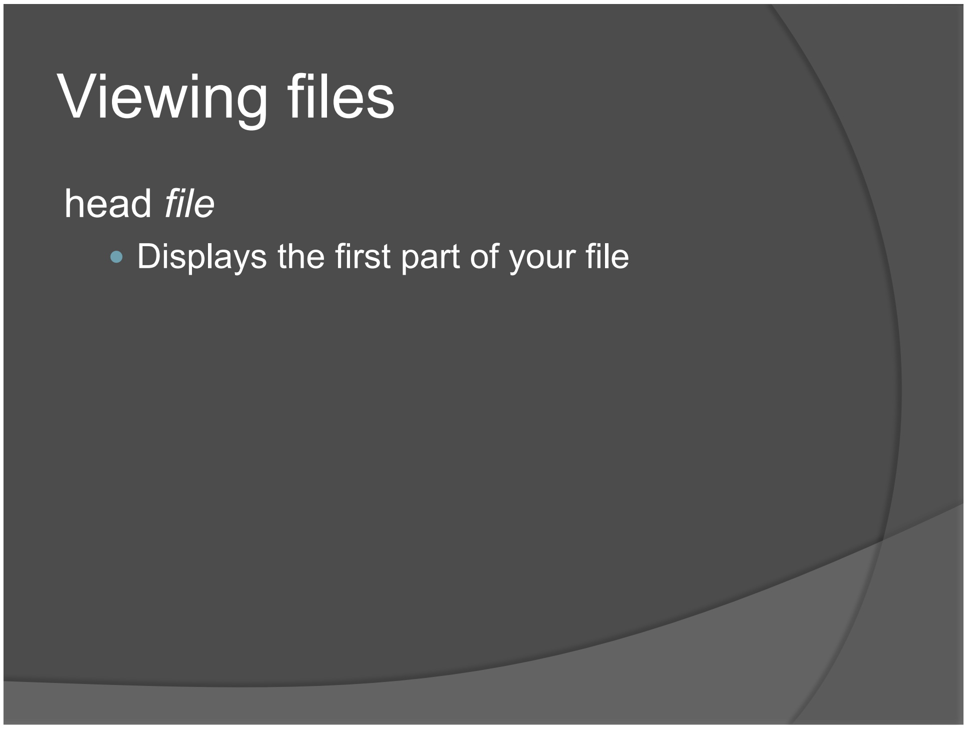
- Lets you page through your file
- Space bar to page down
- Type b to go back or page up
- Type q to quit



Viewing files

less *file*

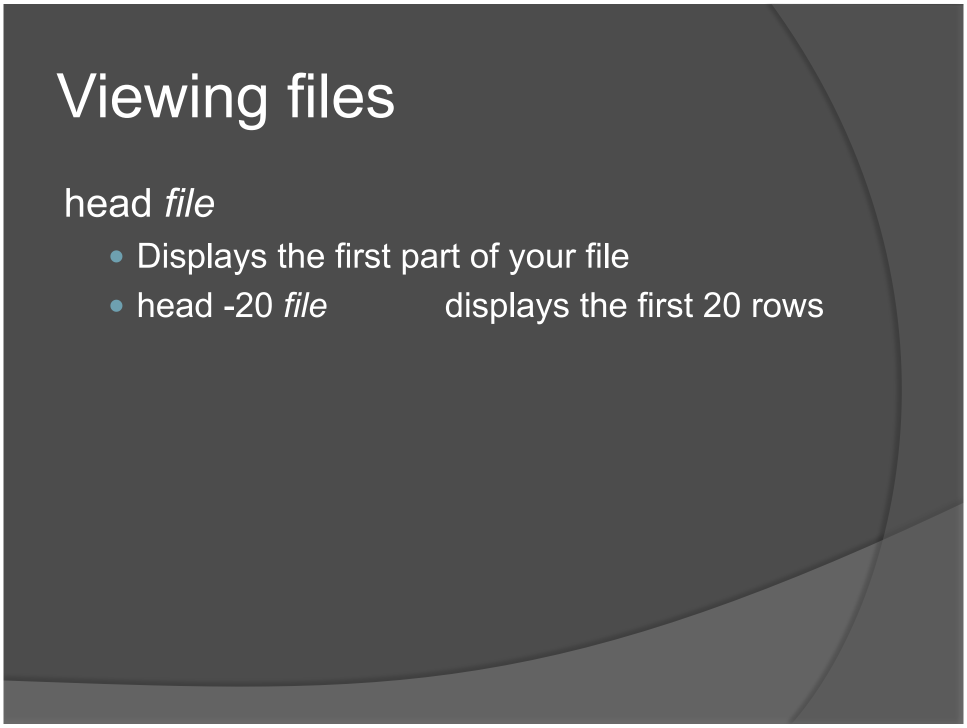
- Basically the same as more



Viewing files

head *file*

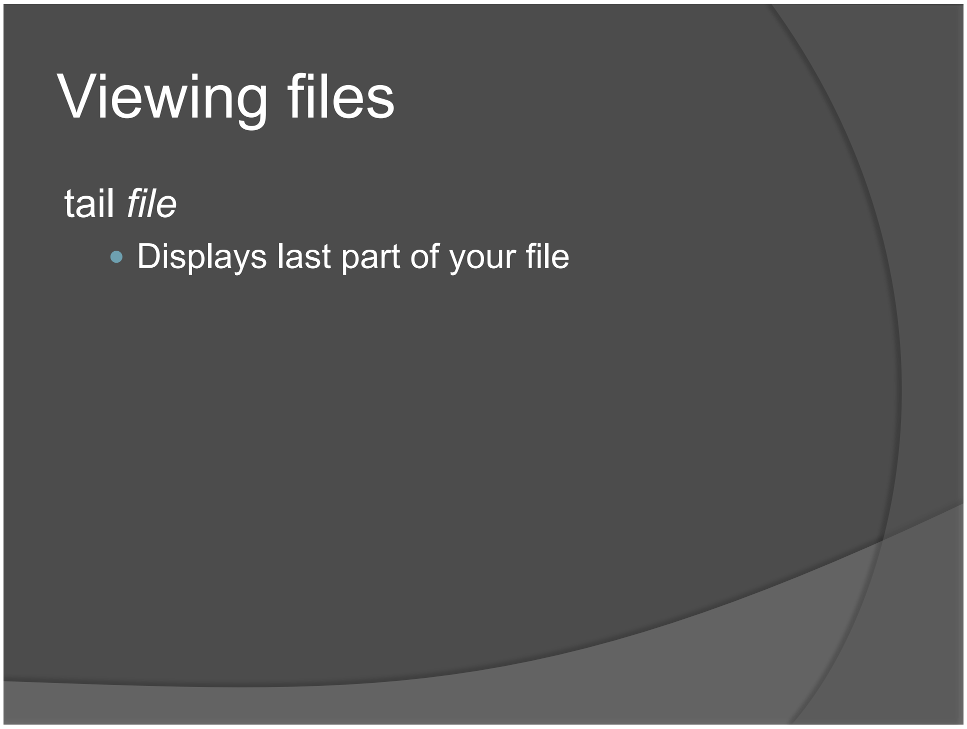
- Displays the first part of your file



Viewing files

head *file*

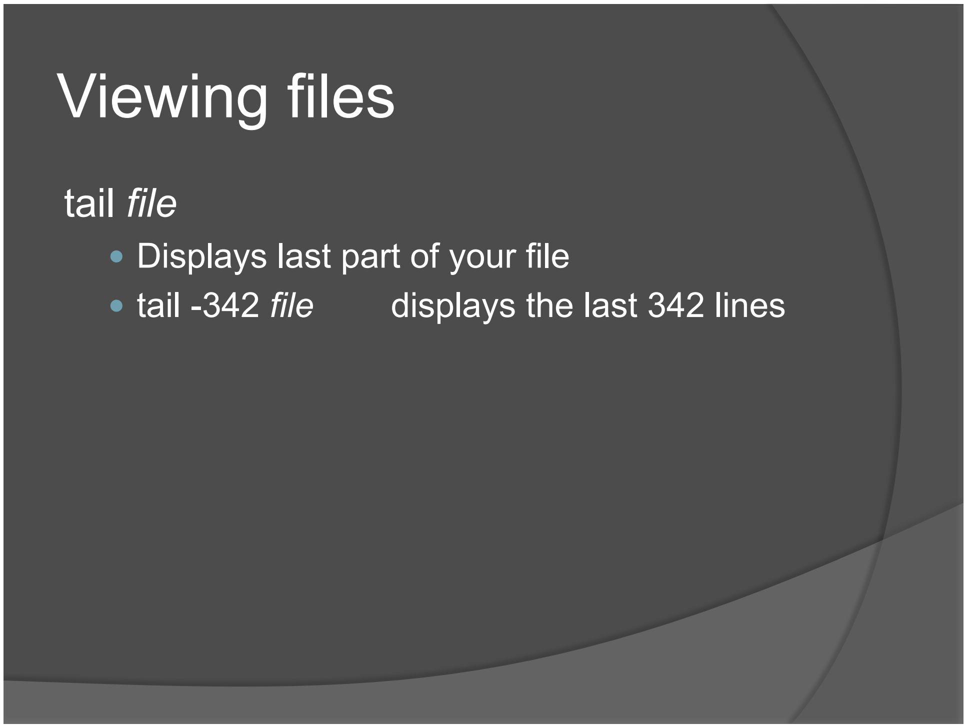
- Displays the first part of your file
- head -20 *file* displays the first 20 rows



Viewing files

tail *file*

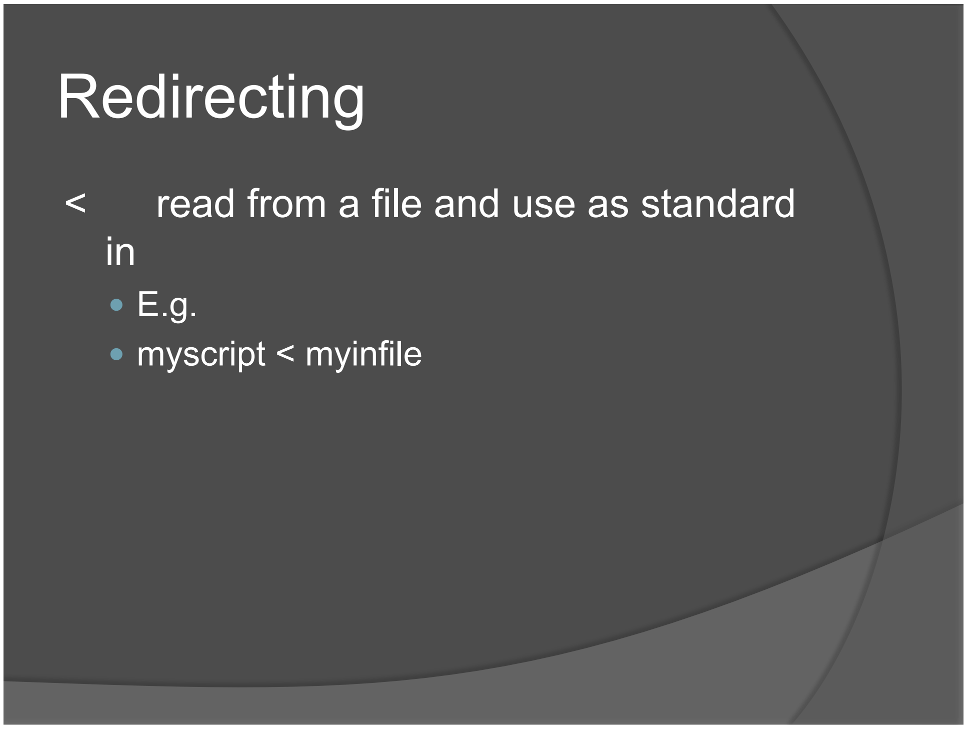
- Displays last part of your file



Viewing files

tail *file*

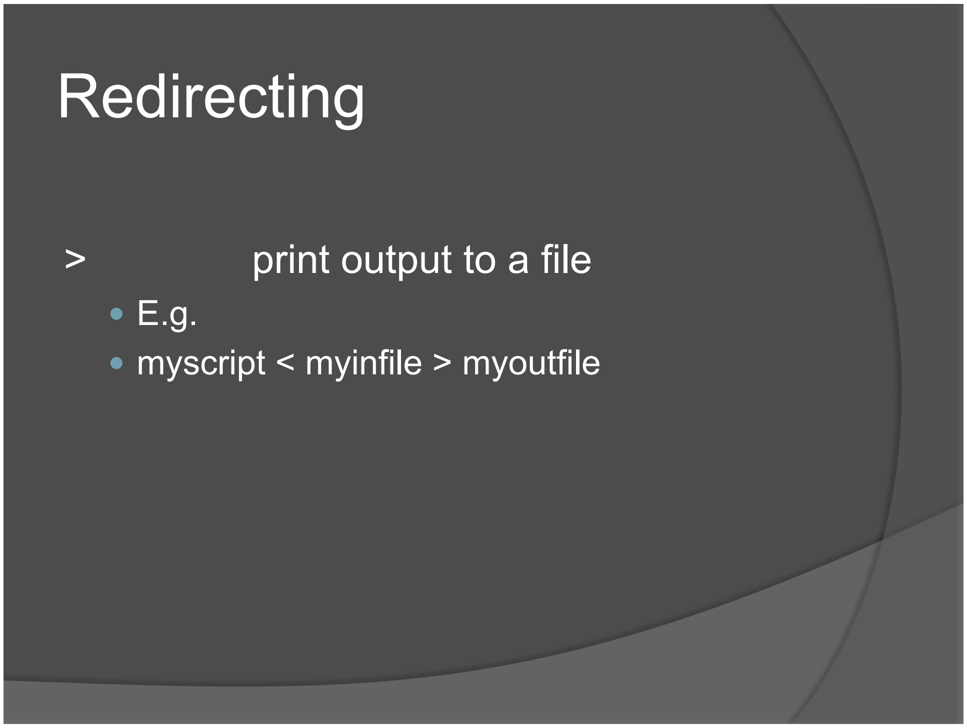
- Displays last part of your file
- tail -342 *file*   displays the last 342 lines



Redirecting

<   read from a file and use as standard in

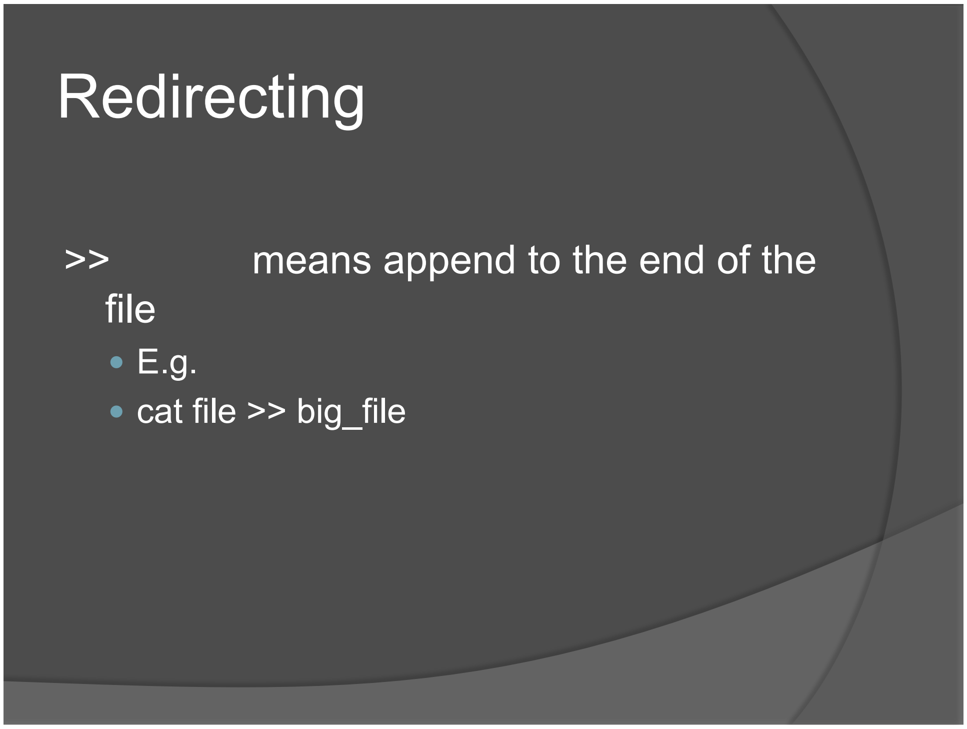
- E.g.
- myscript < myinfile



Redirecting

> print output to a file

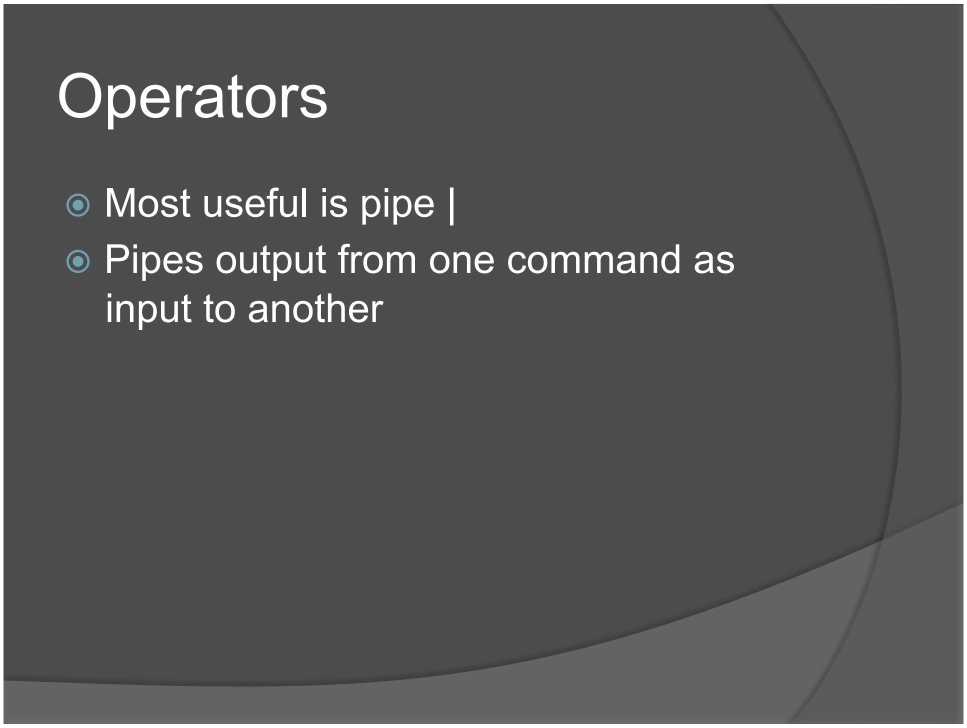
- E.g.
- myscript < myinfile > myoutfile



Redirecting

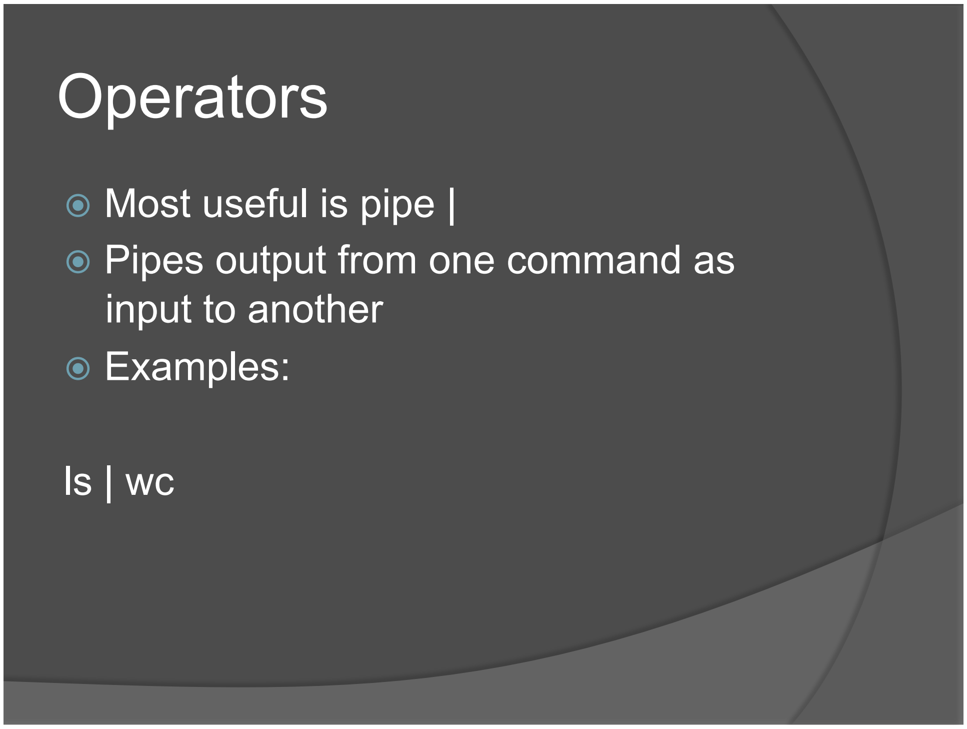
>> means append to the end of the file

- E.g.
- cat file >> big_file



Operators

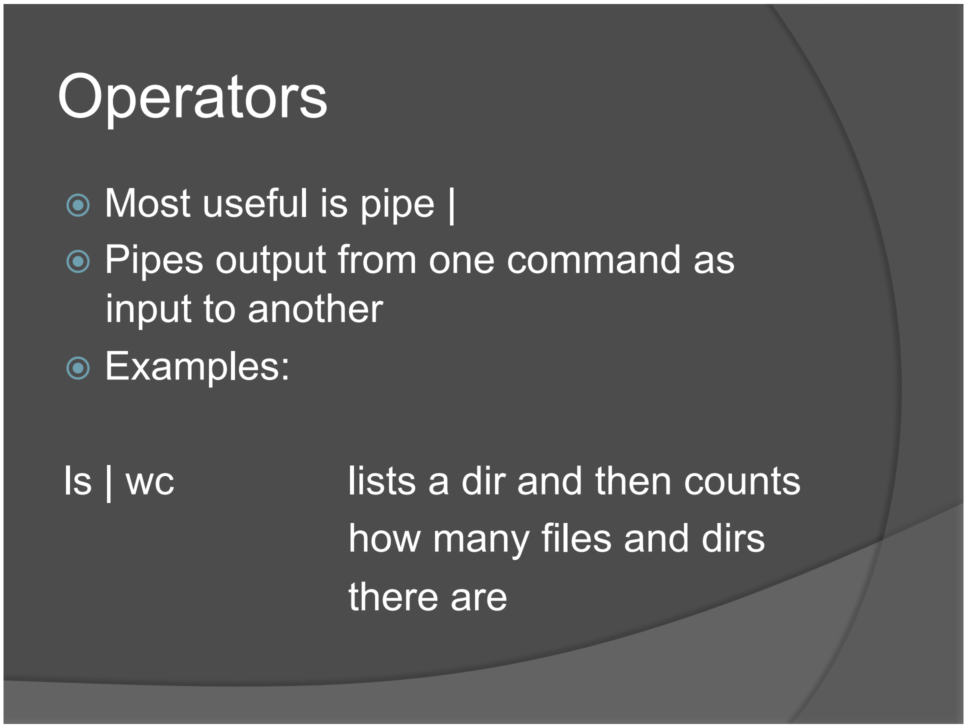
- Most useful is pipe |
- Pipes output from one command as input to another



Operators

- Most useful is pipe |
- Pipes output from one command as input to another
- Examples:

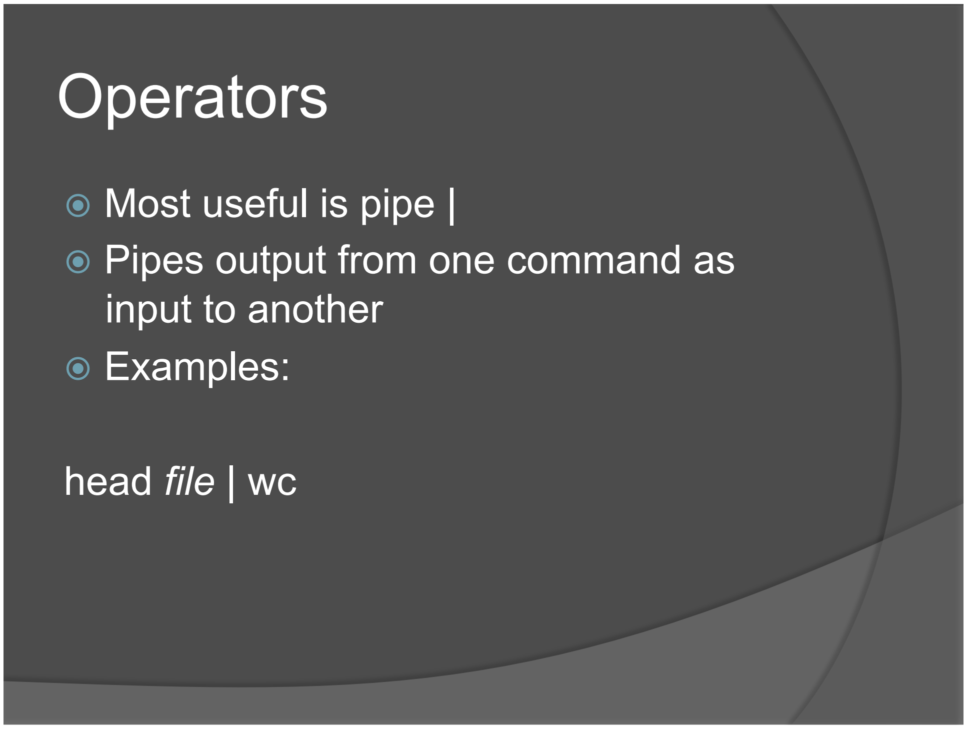
```
ls | wc
```



Operators

- Most useful is pipe |
- Pipes output from one command as input to another
- Examples:

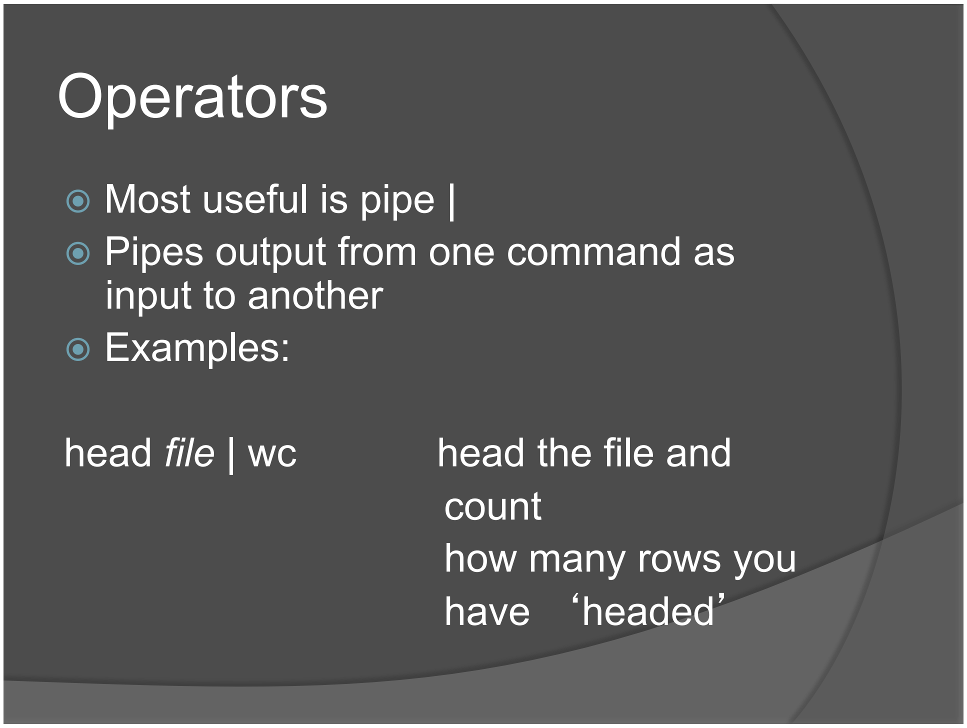
ls | wc	lists a dir and then counts how many files and dirs there are



Operators

- Most useful is pipe |
- Pipes output from one command as input to another
- Examples:

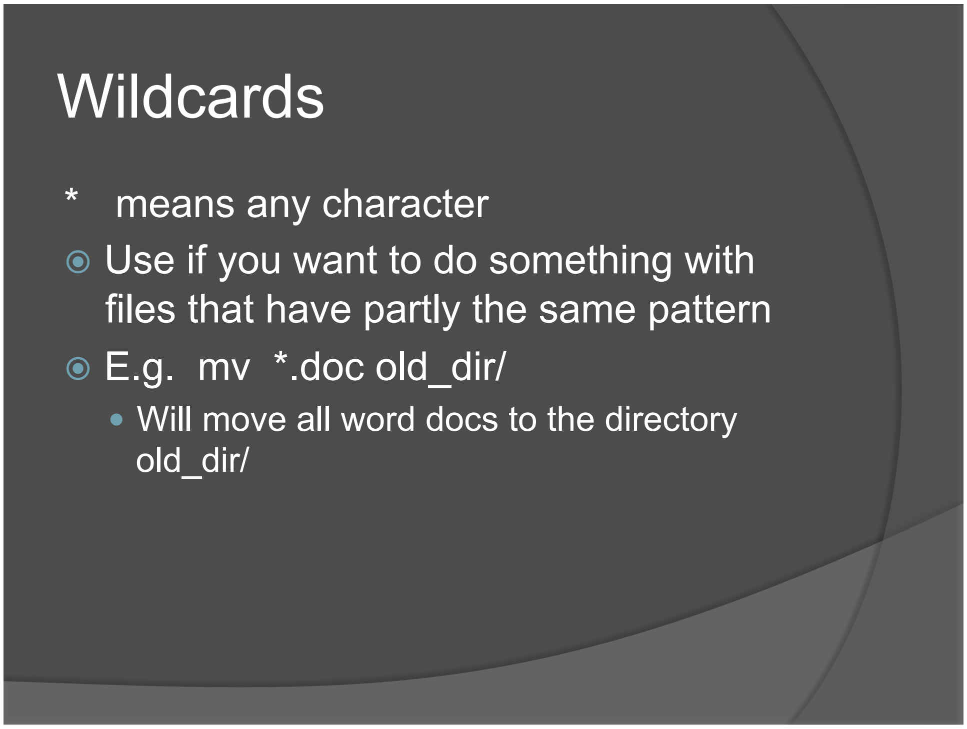
head *file* | wc



Operators

- Most useful is pipe |
- Pipes output from one command as input to another
- Examples:

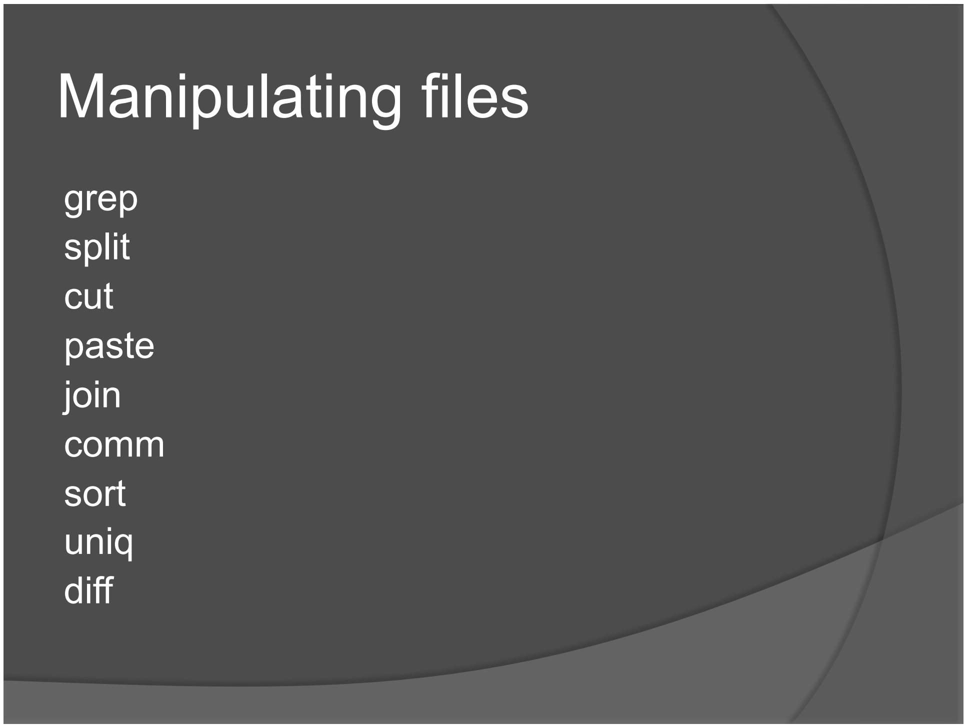
head *file* | wc head the file and count how many rows you have ‘headed’



Wildcards

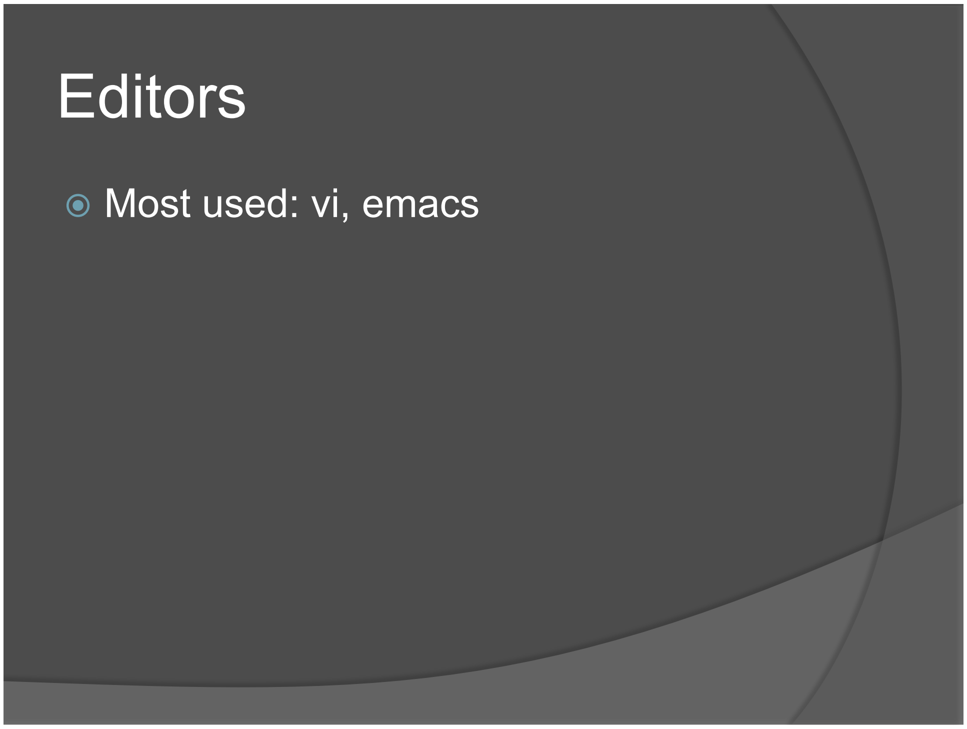
* means any character

- Use if you want to do something with files that have partly the same pattern
- E.g. mv *.doc old_dir/
  - Will move all word docs to the directory old_dir/



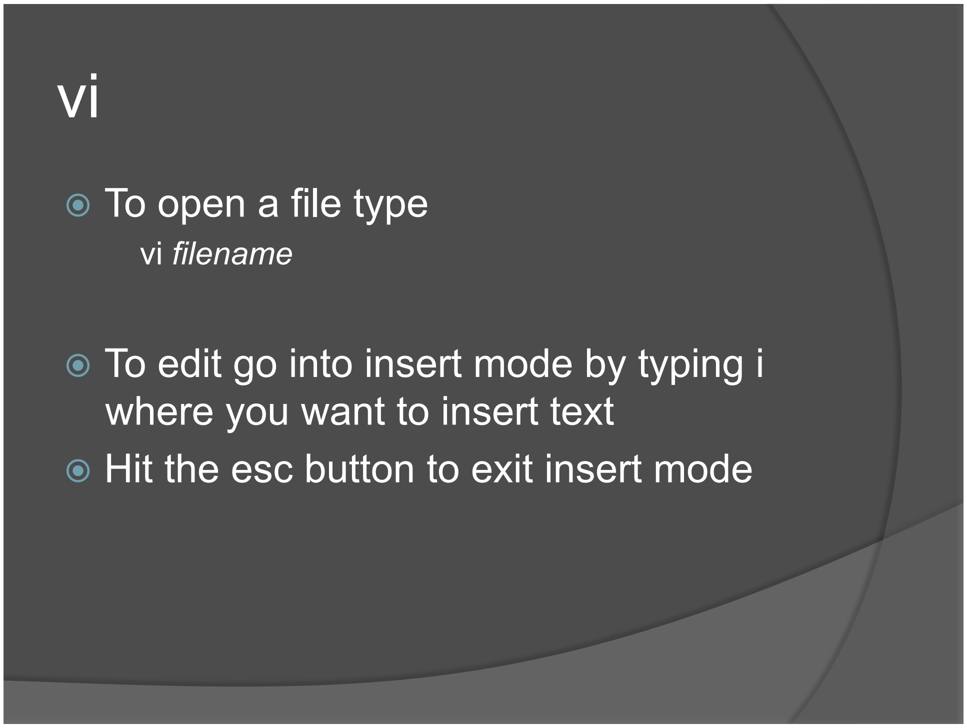
Manipulating files

grep
split
cut
paste
join
comm
sort
uniq
diff



Editors

- Most used: vi, emacs

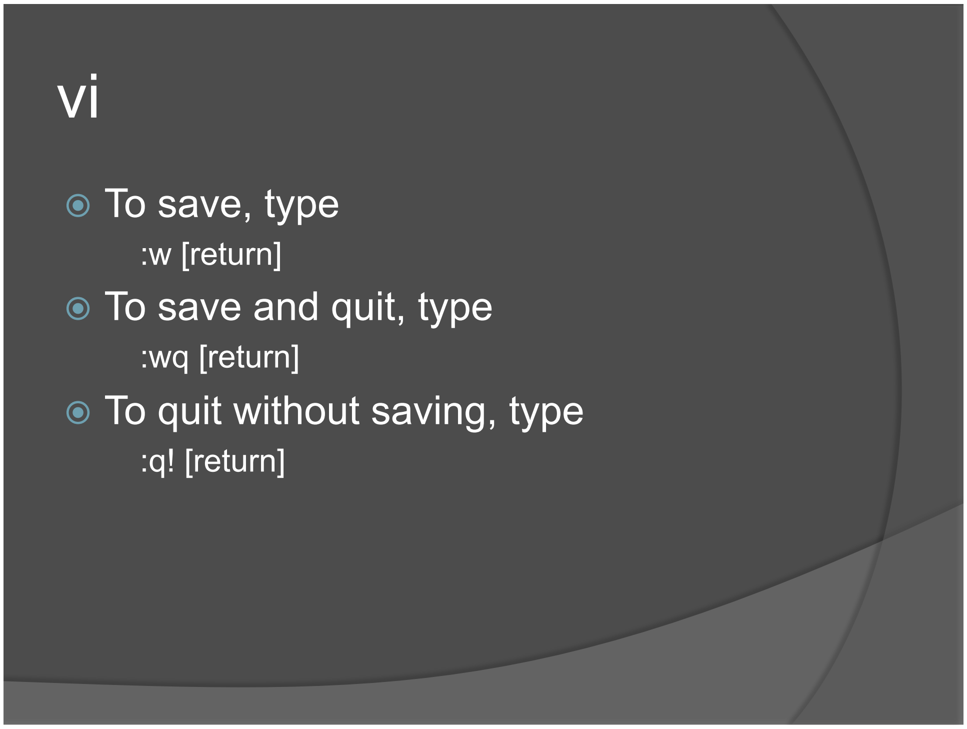


vi

- To open a file type

  vi *filename*

- To edit go into insert mode by typing i where you want to insert text
- Hit the esc button to exit insert mode



vi

- To save, type

  :w [return]

- To save and quit, type

  :wq [return]

- To quit without saving, type

  :q! [return]

tar archives

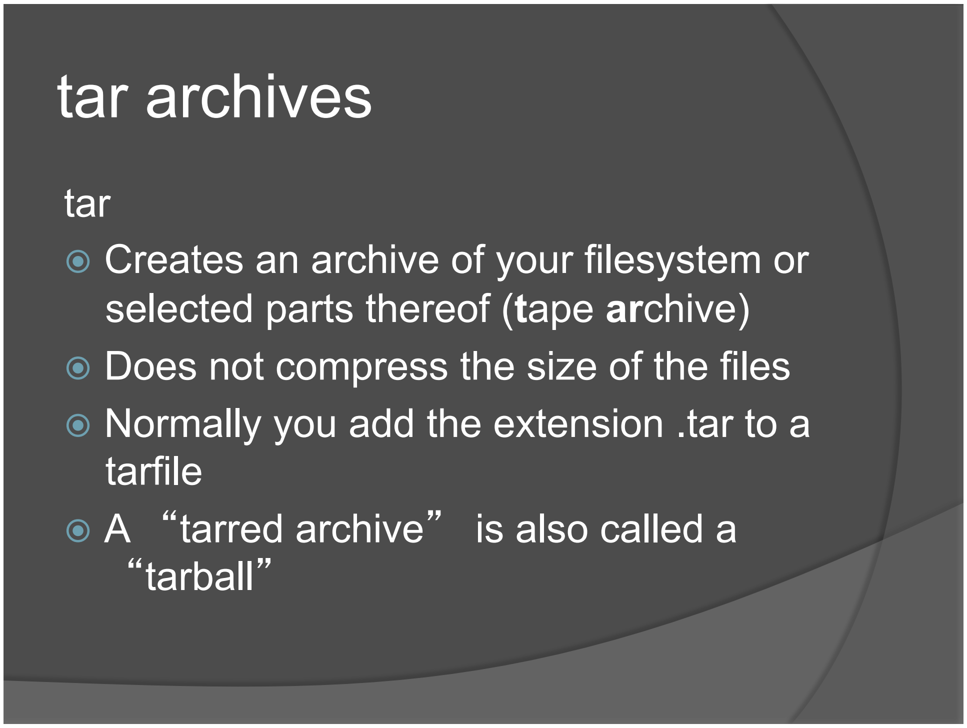
tar

- Creates an archive of your filesystem or selected parts thereof (**t**ape **ar**chive)

tar archives

tar

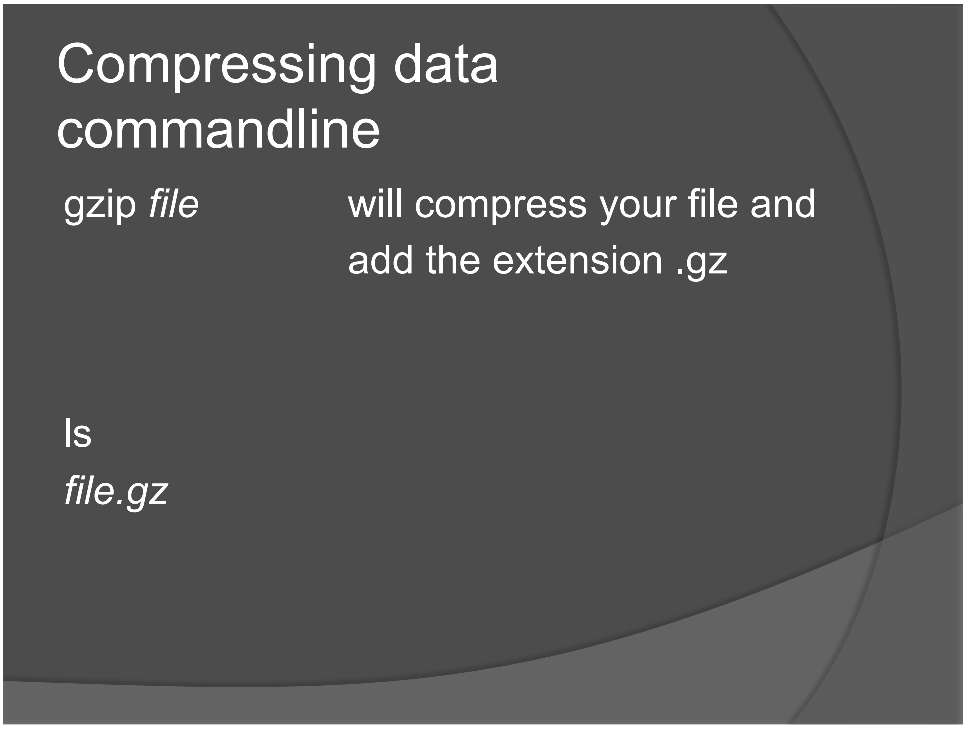
- Creates an archive of your filesystem or selected parts thereof (**t**ape **ar**chive)
- Does not compress the size of the files



tar archives

tar

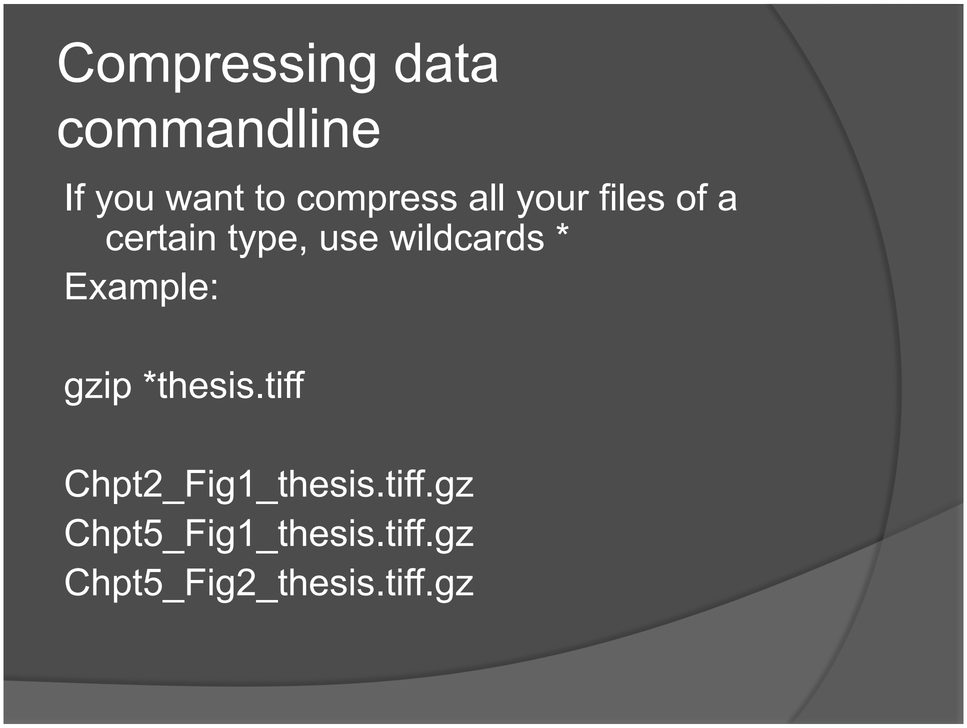
- Creates an archive of your filesystem or selected parts thereof (**t**ape **ar**chive)
- Does not compress the size of the files
- Normally you add the extension .tar to a tarfile
- A "tarred archive" is also called a "tarball"



Compressing data commandline

gzip *file* will compress your file and add the extension .gz

```
ls
file.gz
```



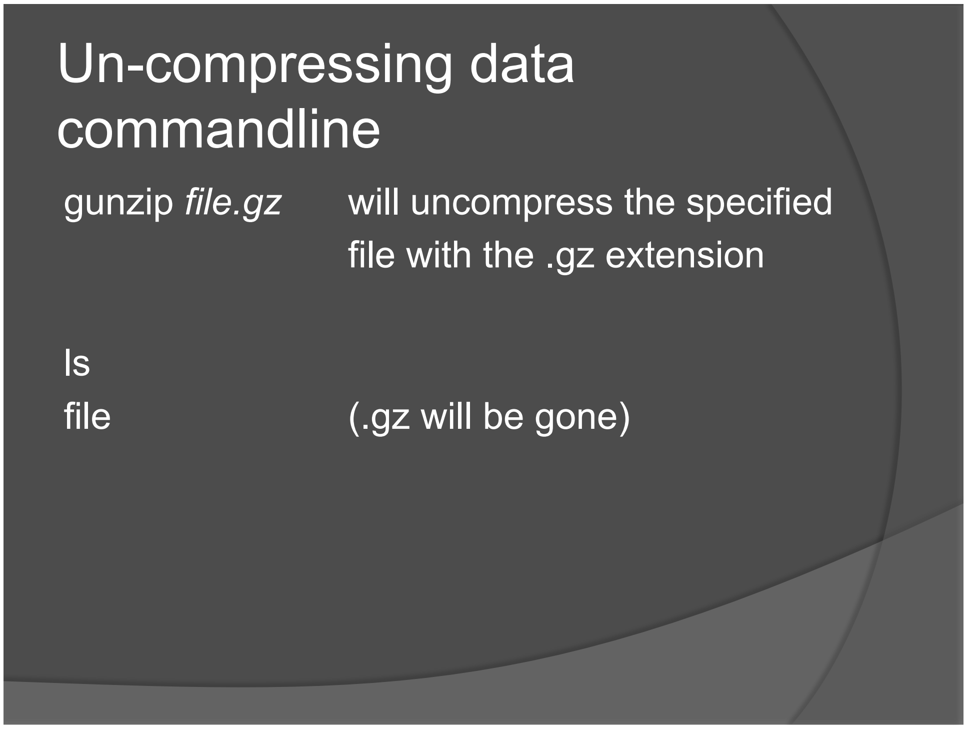
Compressing data commandline

If you want to compress all your files of a certain type, use wildcards *

Example:

```
gzip *thesis.tiff
```

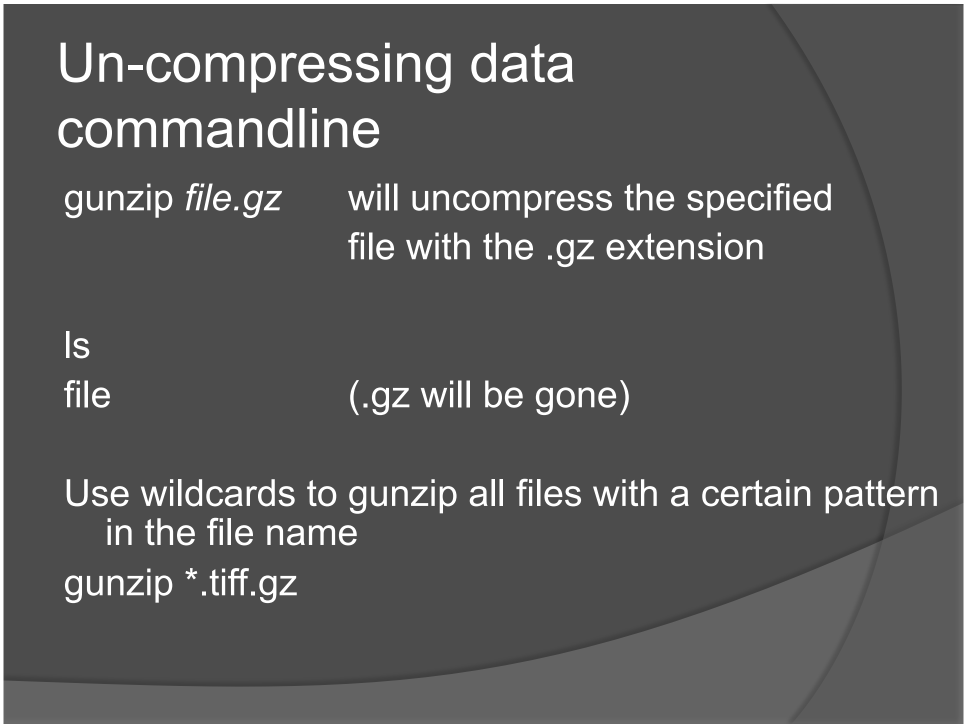
```
Chpt2_Fig1_thesis.tiff.gz
Chpt5_Fig1_thesis.tiff.gz
Chpt5_Fig2_thesis.tiff.gz
```



Un-compressing data commandline

gunzip *file.gz*   will uncompress the specified file with the .gz extension

```
ls
file                    (.gz will be gone)
```



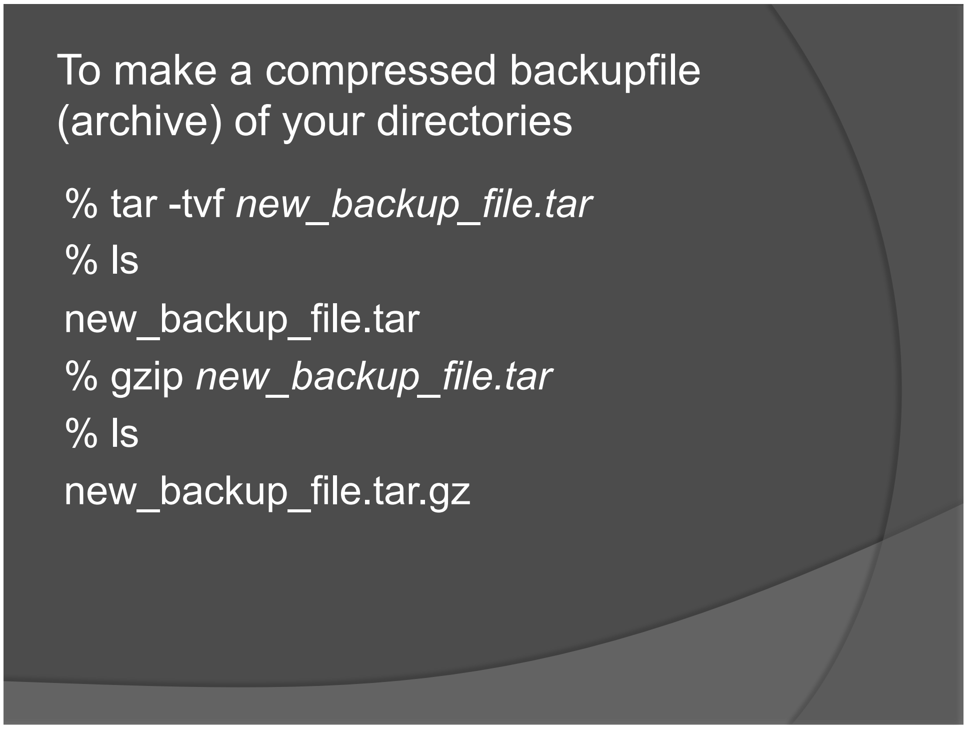
Un-compressing data commandline

gunzip *file.gz* will uncompress the specified file with the .gz extension

```
ls
file                 (.gz will be gone)
```

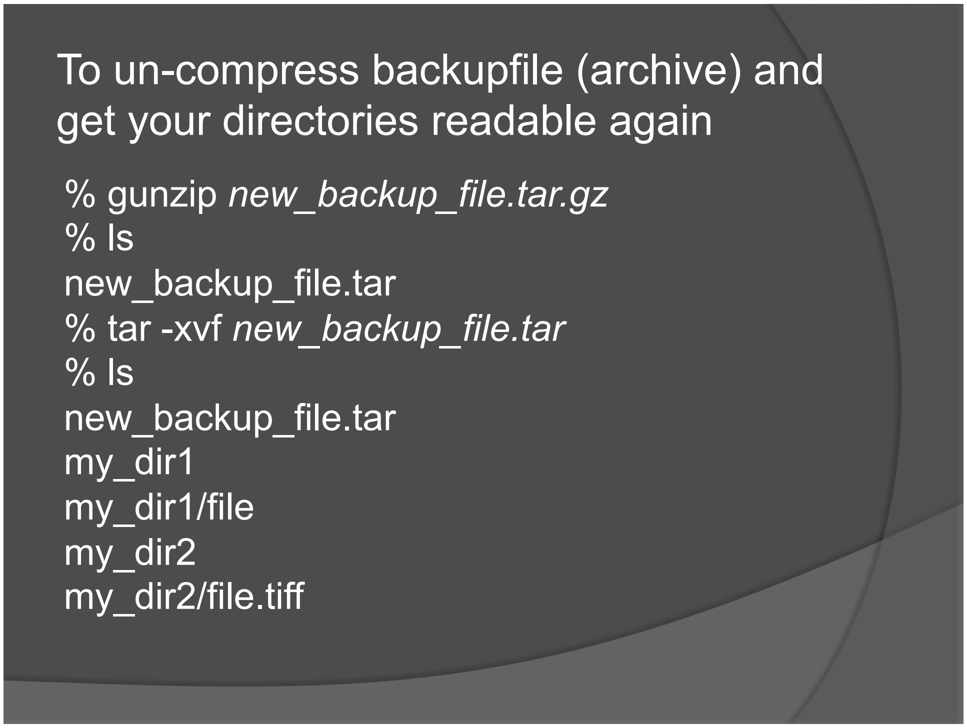
Use wildcards to gunzip all files with a certain pattern in the file name

```
gunzip *.tiff.gz
```



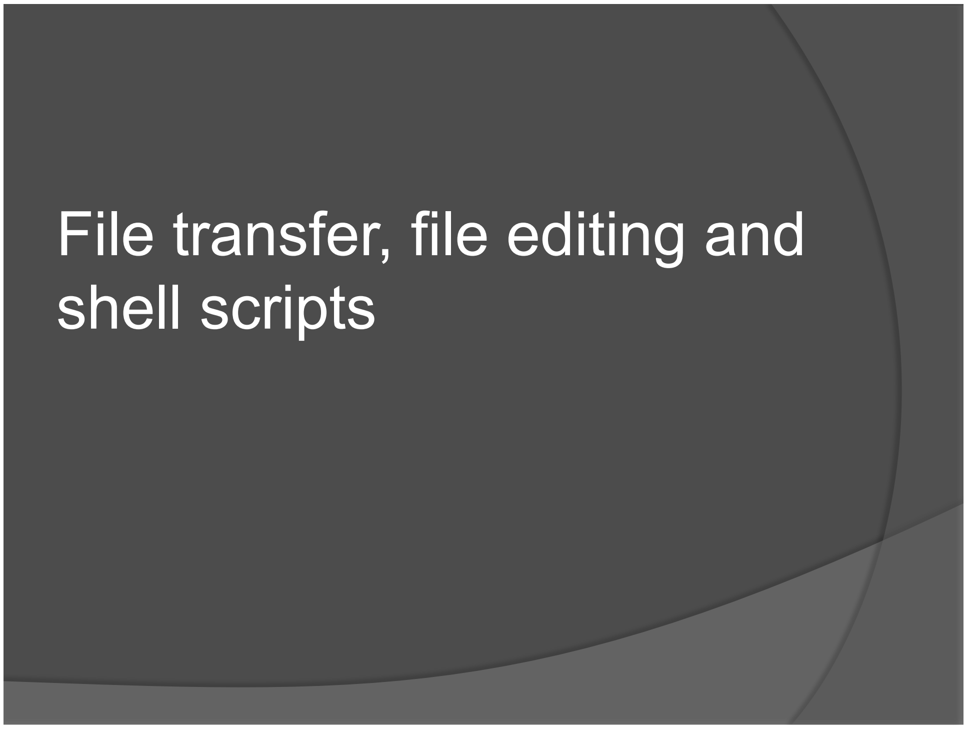
To make a compressed backupfile (archive) of your directories

```
% tar -tvf new_backup_file.tar
% ls
new_backup_file.tar
% gzip new_backup_file.tar
% ls
new_backup_file.tar.gz
```

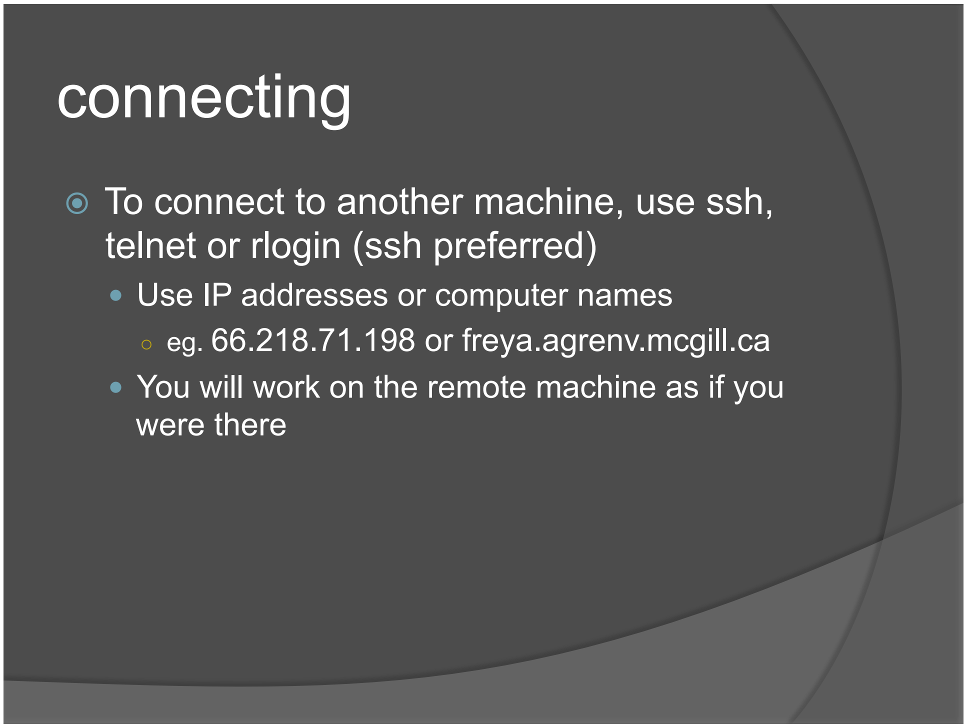


To un-compress backupfile (archive) and get your directories readable again

```
% gunzip new_backup_file.tar.gz
% ls
new_backup_file.tar
% tar -xvf new_backup_file.tar
% ls
new_backup_file.tar
my_dir1
my_dir1/file
my_dir2
my_dir2/file.tiff
```

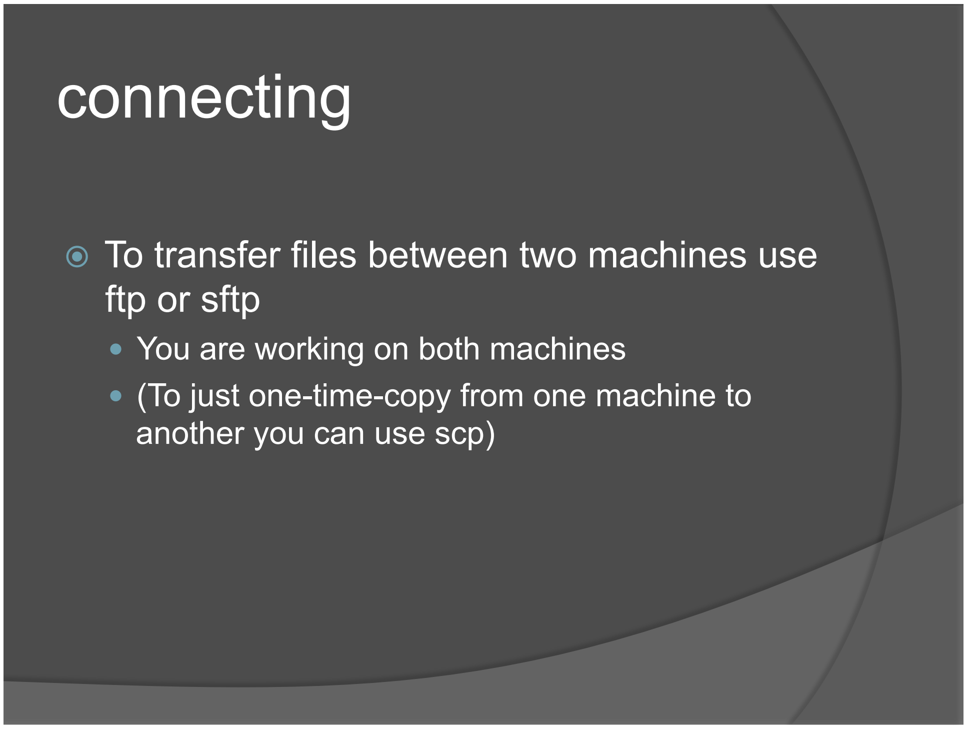


File transfer, file editing and shell scripts



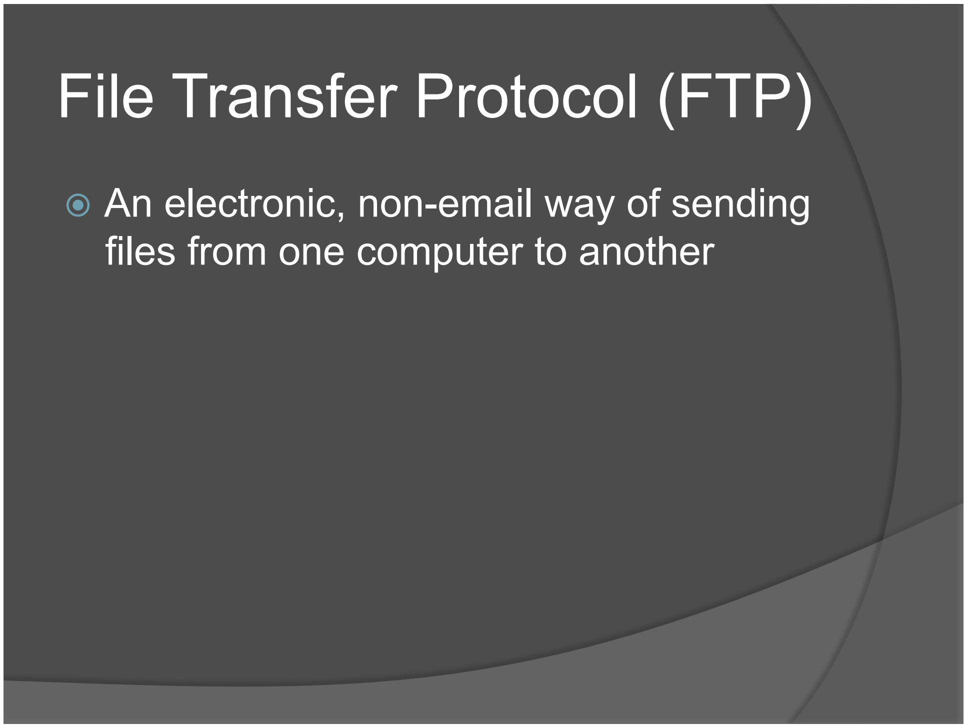
connecting

- To connect to another machine, use ssh, telnet or rlogin (ssh preferred)
  - Use IP addresses or computer names
    - eg. 66.218.71.198 or freya.agrenv.mcgill.ca
  - You will work on the remote machine as if you were there



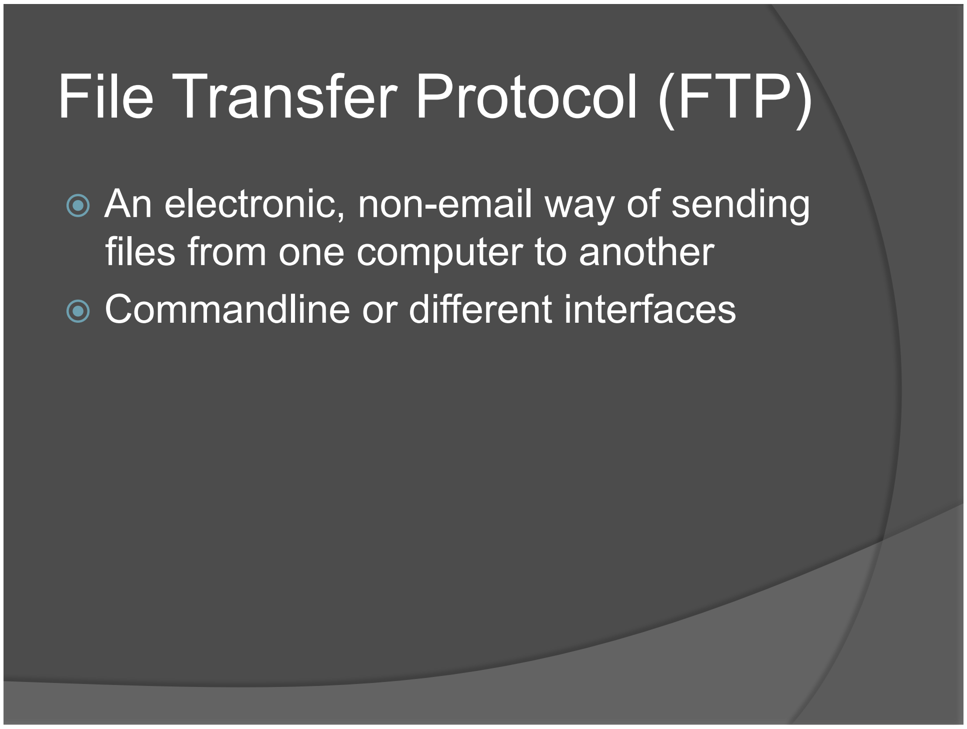
connecting

- To transfer files between two machines use ftp or sftp
  - You are working on both machines
  - (To just one-time-copy from one machine to another you can use scp)



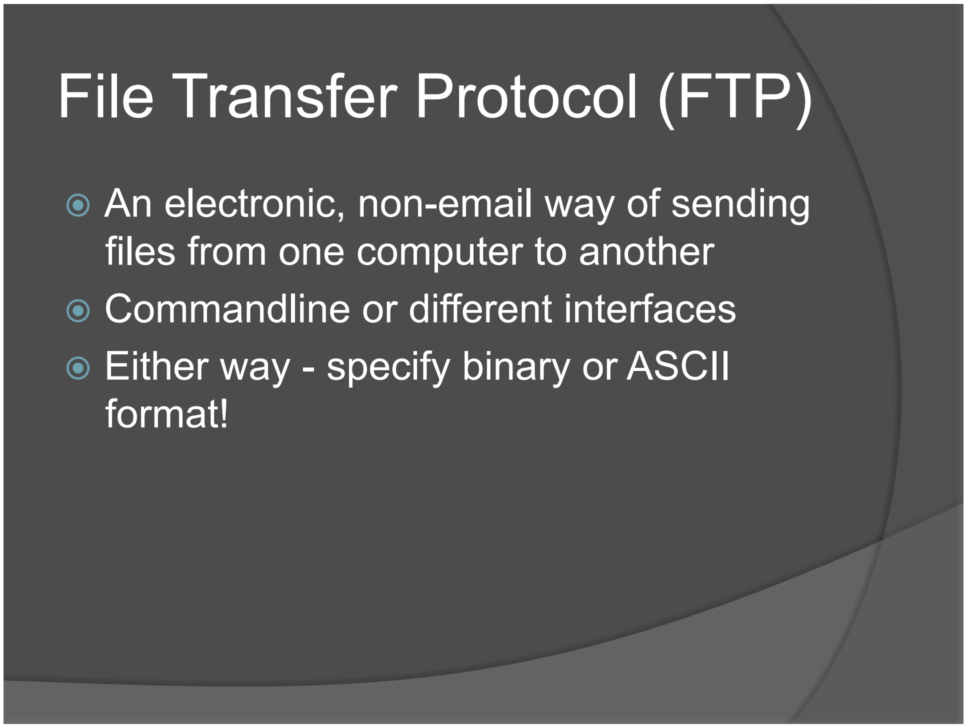
File Transfer Protocol (FTP)

- An electronic, non-email way of sending files from one computer to another



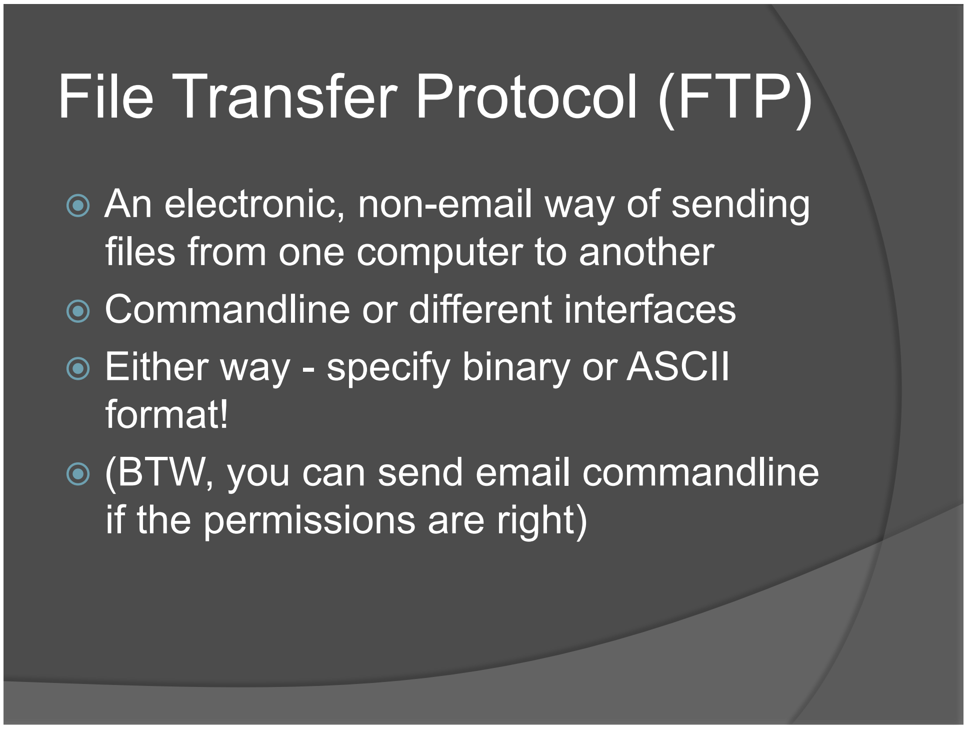
File Transfer Protocol (FTP)

- An electronic, non-email way of sending files from one computer to another
- Commandline or different interfaces



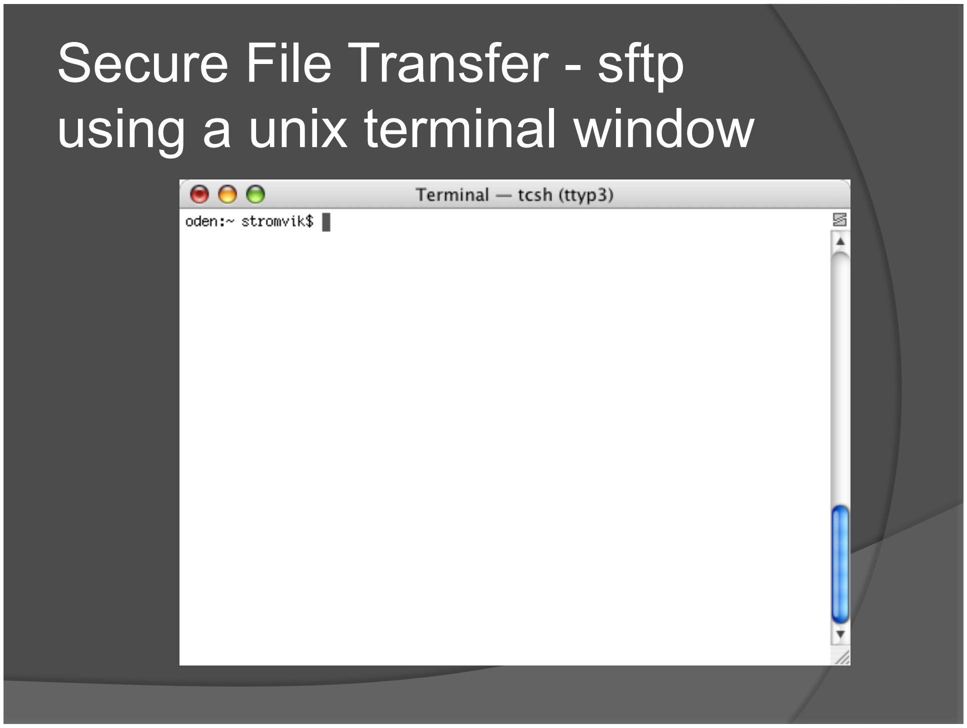
File Transfer Protocol (FTP)

- An electronic, non-email way of sending files from one computer to another
- Commandline or different interfaces
- Either way - specify binary or ASCII format!



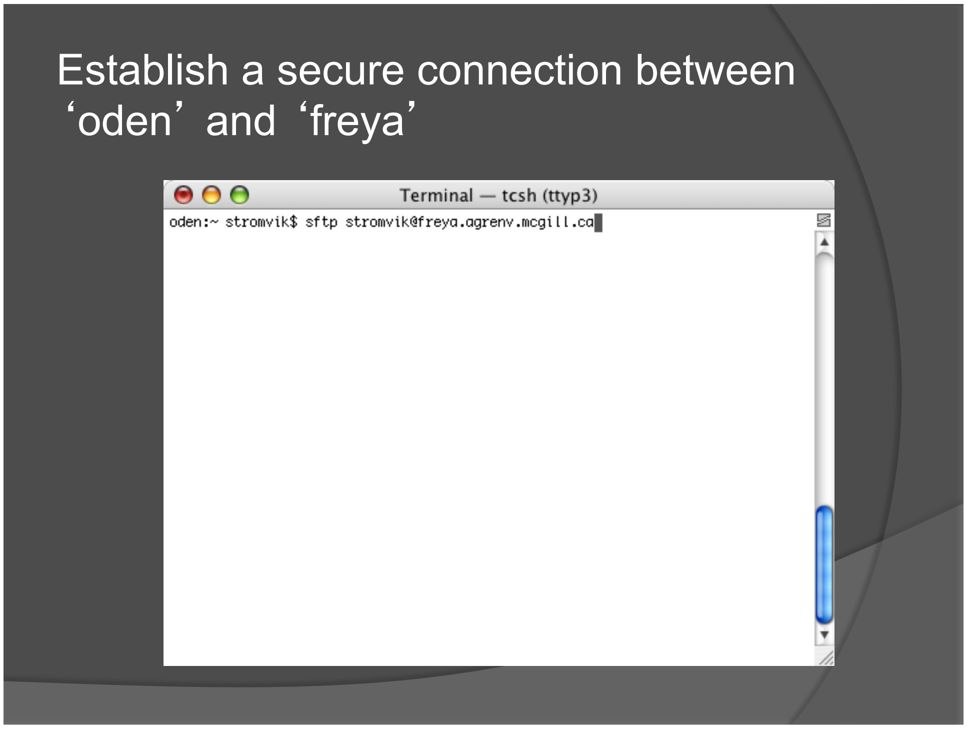
File Transfer Protocol (FTP)

- An electronic, non-email way of sending files from one computer to another
- Commandline or different interfaces
- Either way - specify binary or ASCII format!
- (BTW, you can send email commandline if the permissions are right)



Secure File Transfer - sftp using a unix terminal window

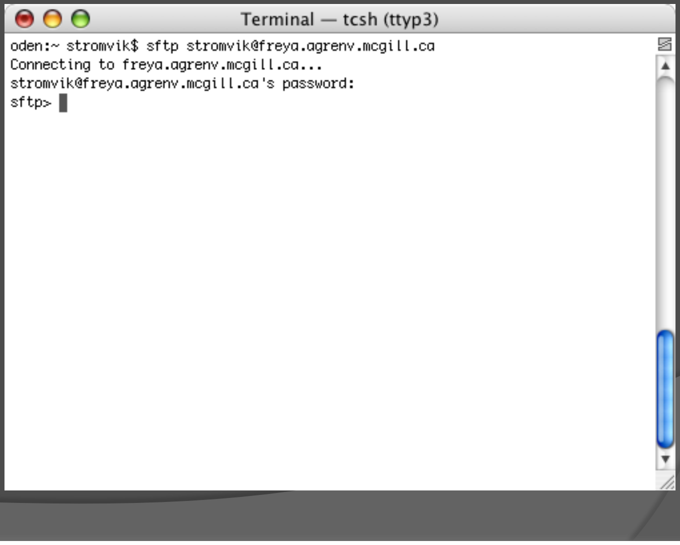
```
oden:~ stromvik$
```



Establish a secure connection between ‘oden’ and ‘freya’

```
oden:~ stromvik$ sftp stromvik@freya.agrenv.mcgill.ca
```

You now talk with Both machines using the program sftp



```
oden:~ stromvik$ sftp stromvik@freya.agrenv.mcgill.ca
Connecting to freya.agrenv.mcgill.ca...
stromvik@freya.agrenv.mcgill.ca's password:
sftp> █
```

Some commands in common between sftp and unix shell

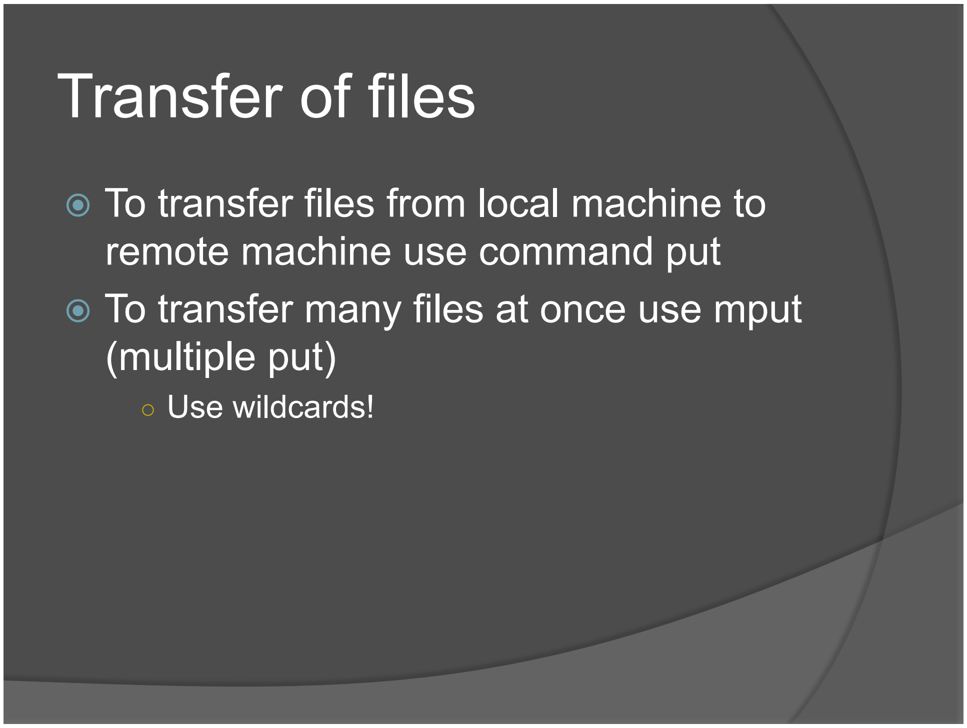
- To list the working dir on remote machine = ls
- To list the working dir on the local machine = lls

Some commands in common with unix

- To print remote working dir = pwd
- To print local working dir = lpwd

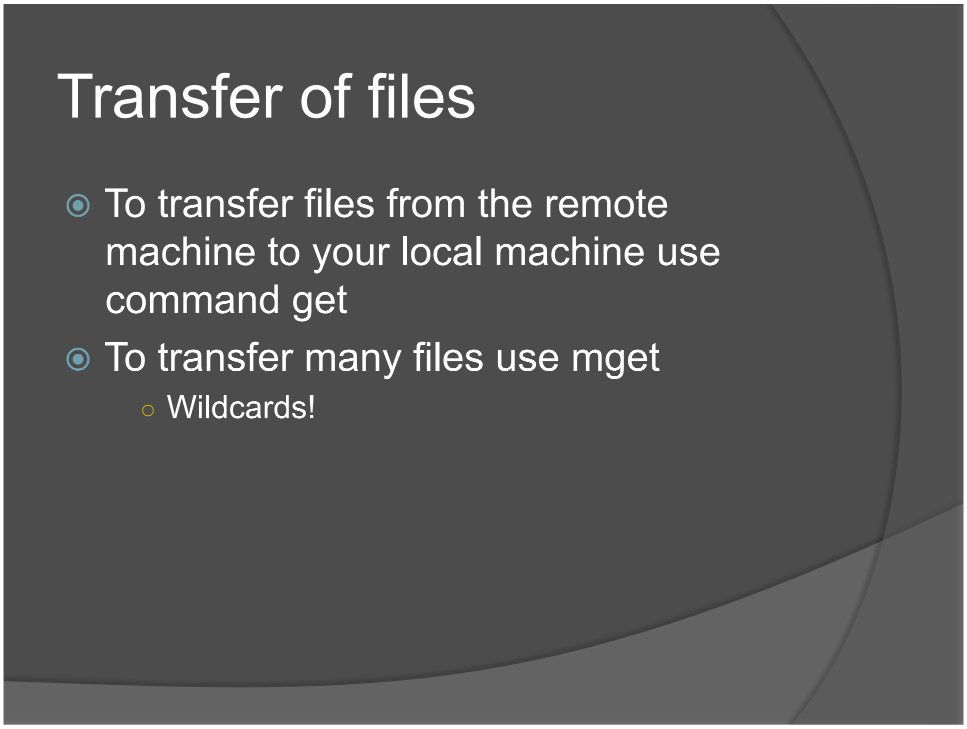
Some commands in common with unix

- To change dir remotely = cd
- To change dir locally = lcd



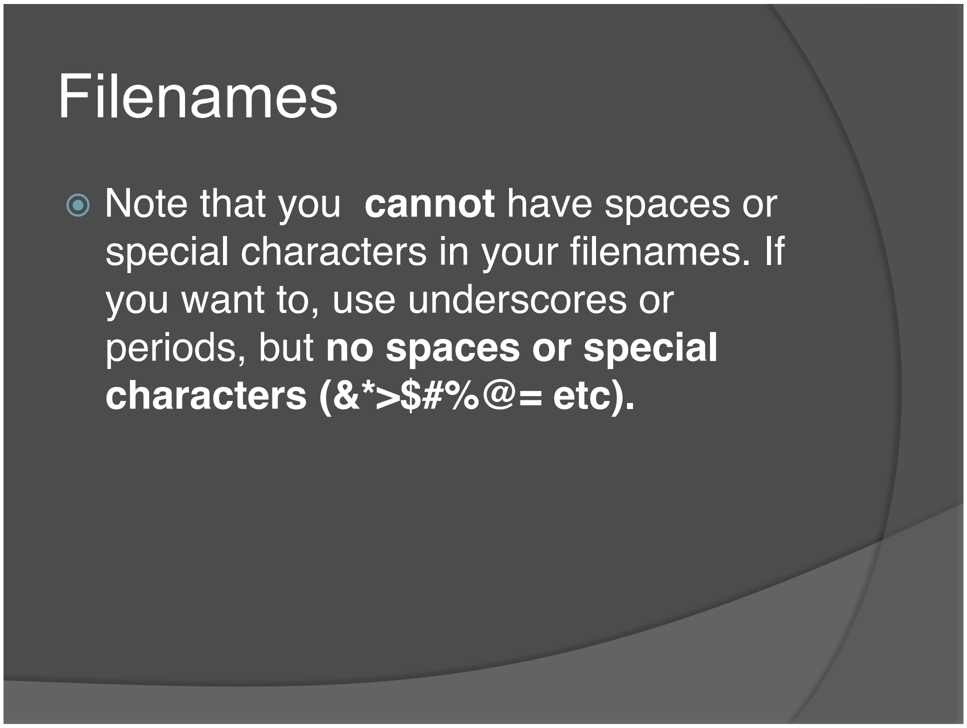
Transfer of files

- To transfer files from local machine to remote machine use command put
- To transfer many files at once use mput (multiple put)
  - Use wildcards!



Transfer of files

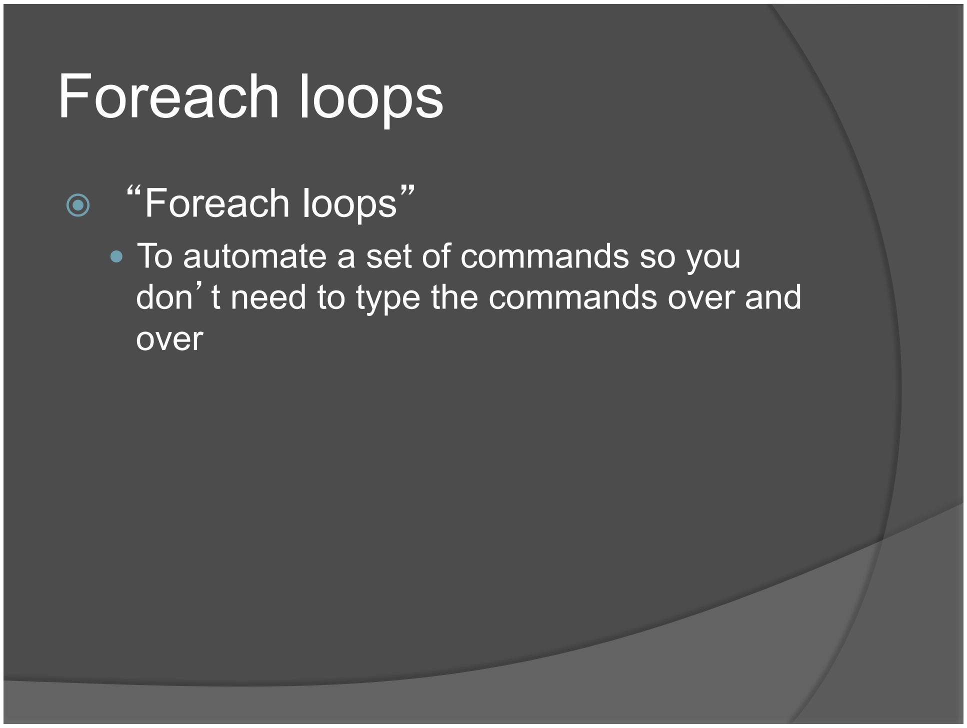
- To transfer files from the remote machine to your local machine use command get
- To transfer many files use mget
  - Wildcards!



Filenames

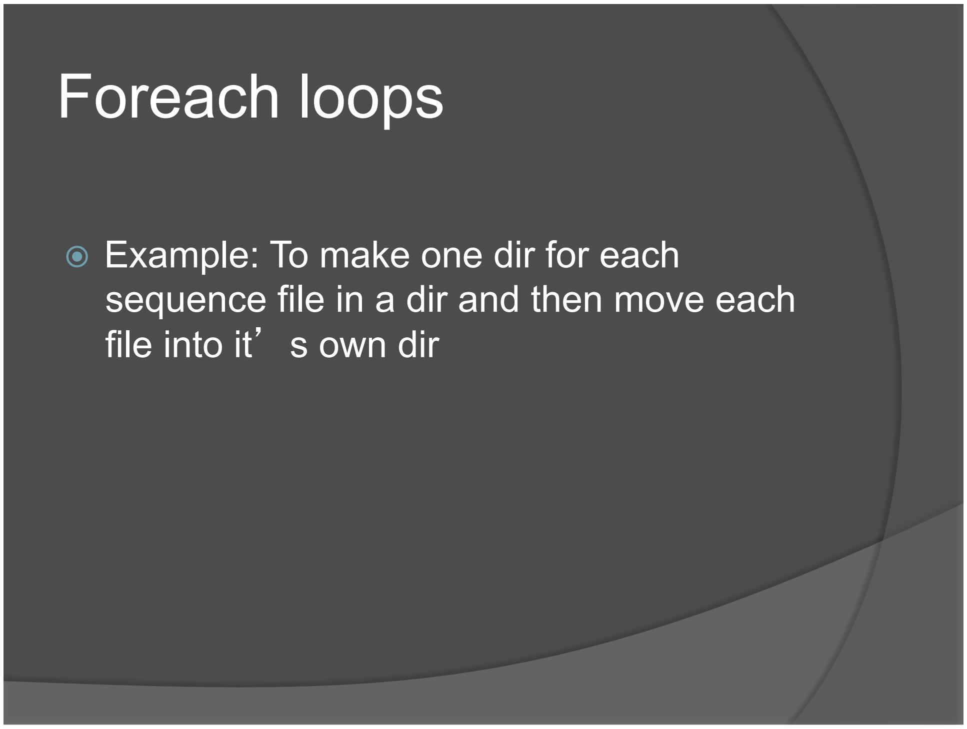
- Note that you **cannot** have spaces or special characters in your filenames. If you want to, use underscores or periods, but **no spaces or special characters (&*>$#%@= etc).**

SHELL SCRIPTS



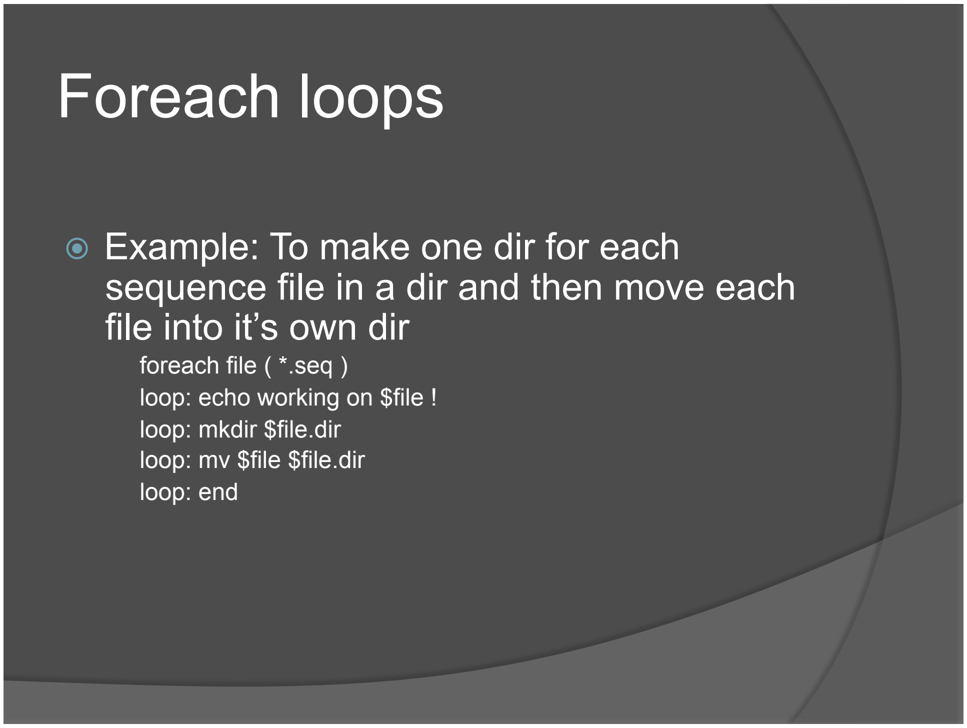
Foreach loops

- "Foreach loops"
  - To automate a set of commands so you don' t need to type the commands over and over



Foreach loops

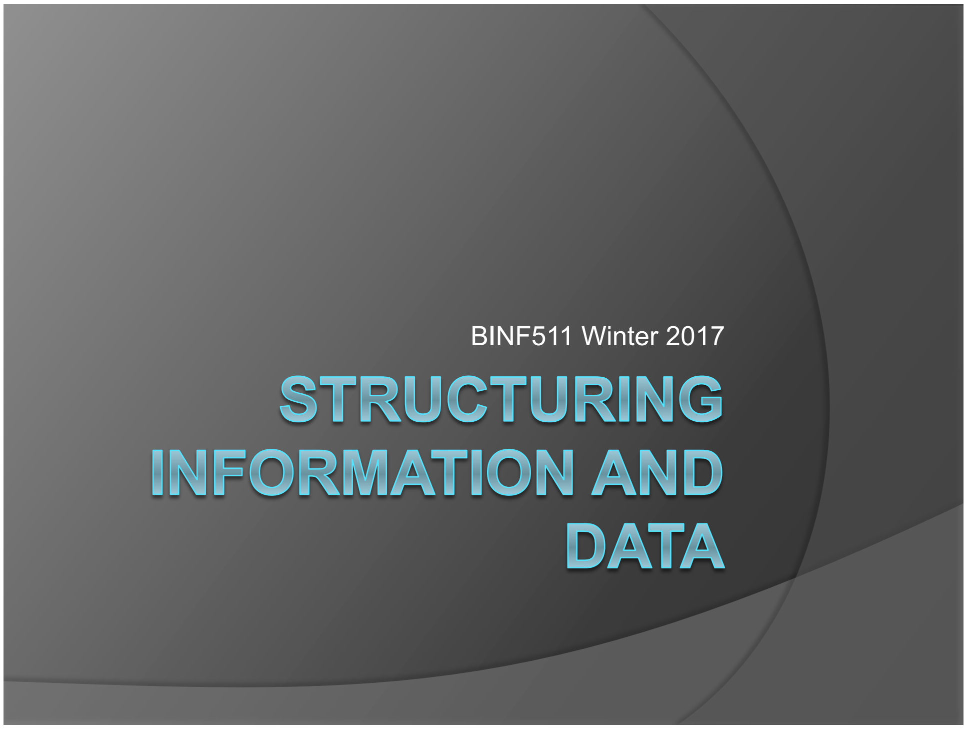
- Example: To make one dir for each sequence file in a dir and then move each file into it' s own dir



Foreach loops

- Example: To make one dir for each sequence file in a dir and then move each file into it’s own dir

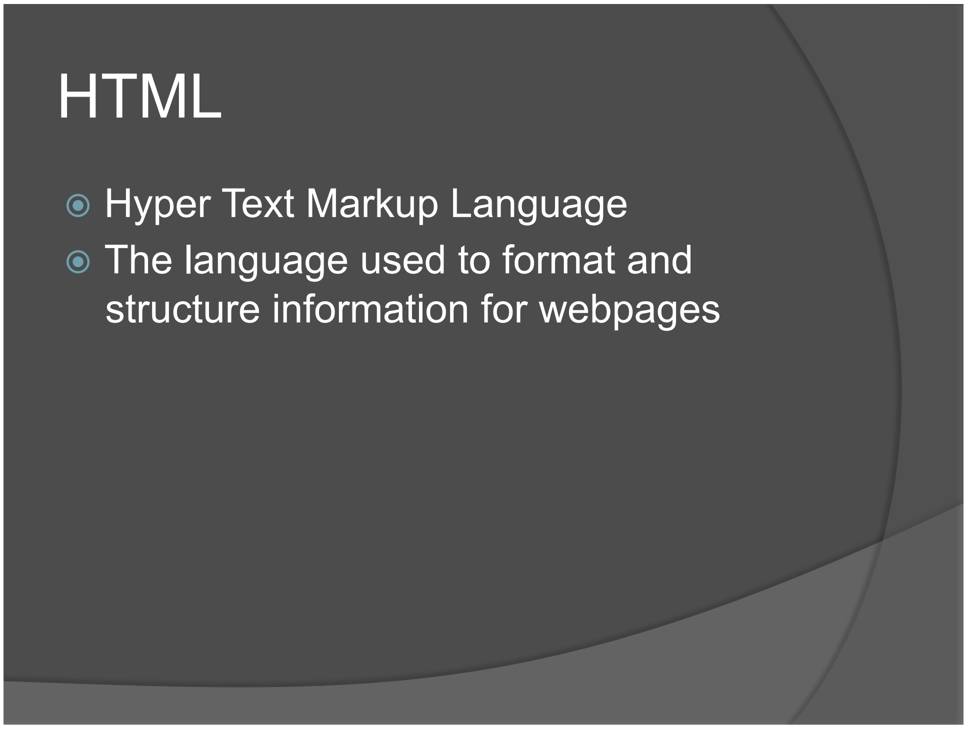
```
foreach file ( *.seq )
loop: echo working on $file !
loop: mkdir $file.dir
loop: mv $file $file.dir
loop: end
```



BINF511 Winter 2017

STRUCTURING INFORMATION AND DATA

HTML



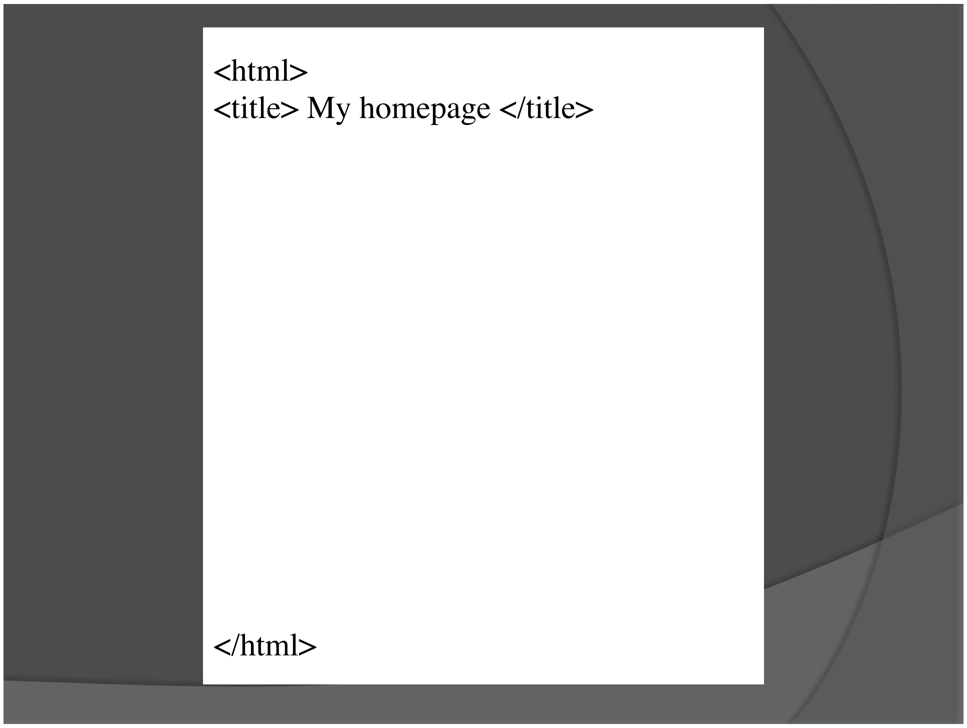
HTML

- Hyper Text Markup Language
- The language used to format and structure information for webpages

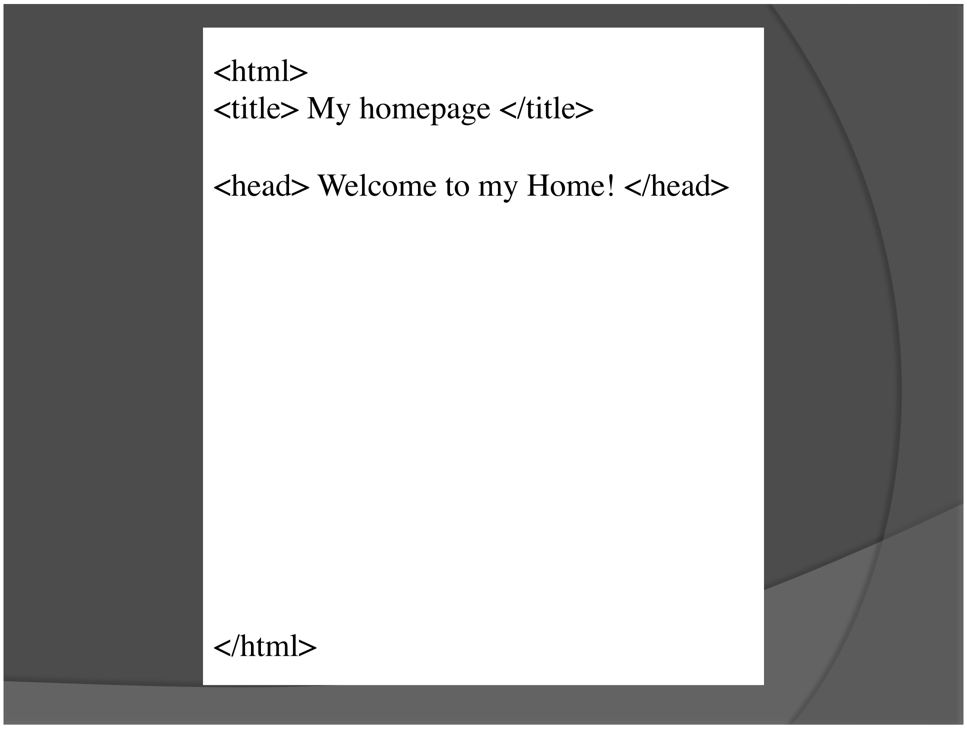
HTML

- Hyper Text Markup Language
- The language used to format and structure information for webpages
- HTML tags - everything is ordered within tags
- Use file suffix .html or .htm
- To interpret, you use a browser
  - Internet Explorer, Netscape, Mozilla, Safari

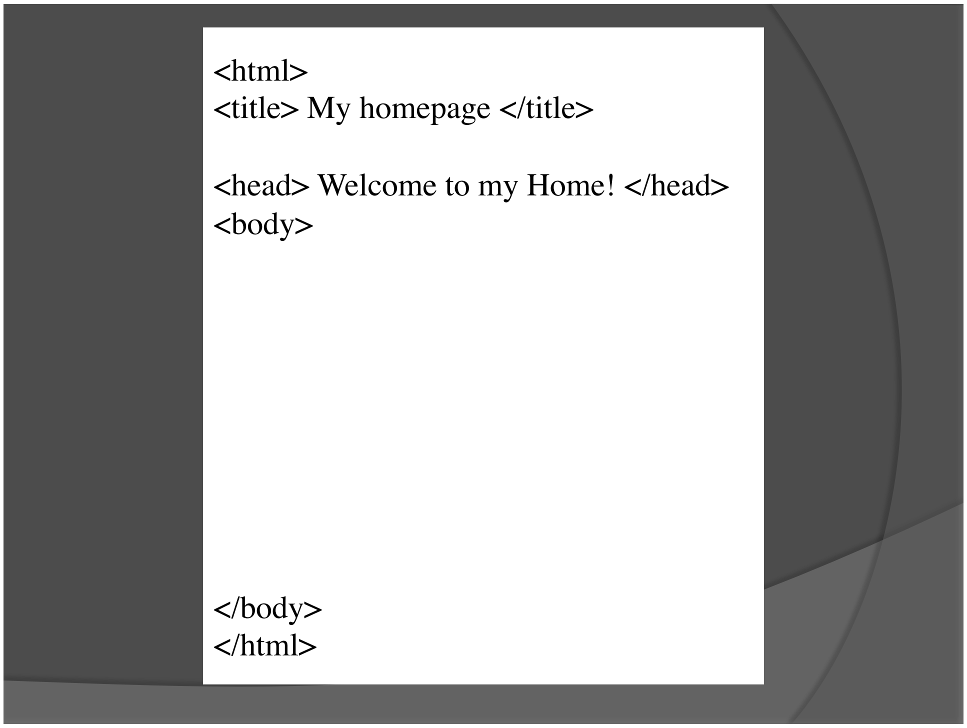
```
<html>



</html>
```



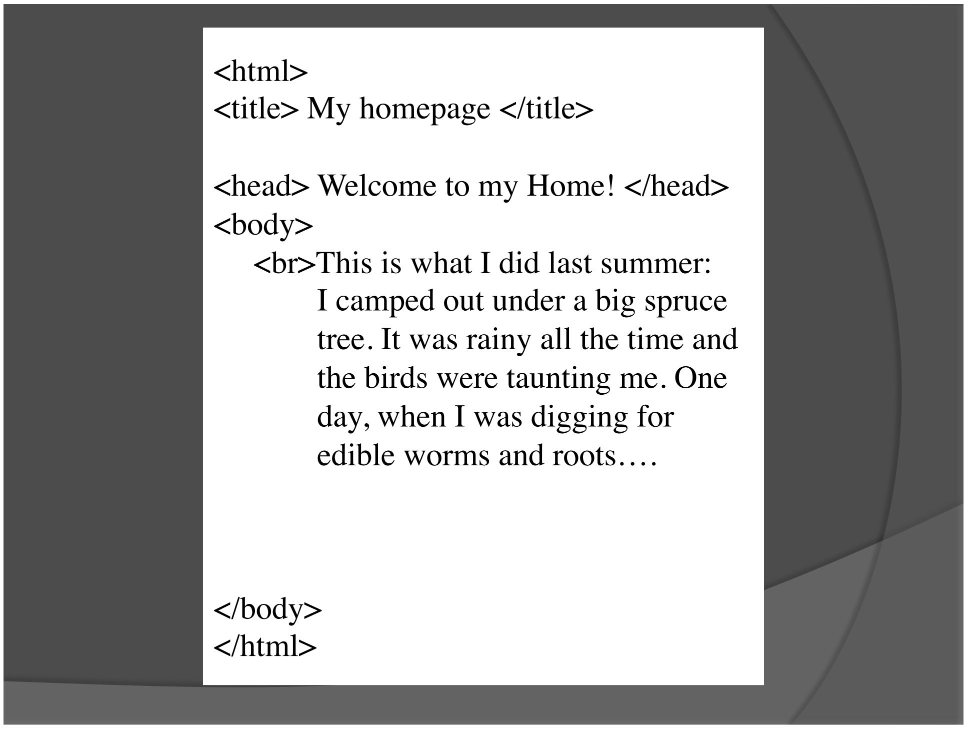
```
<html>
<title> My homepage </title>







</html>
```



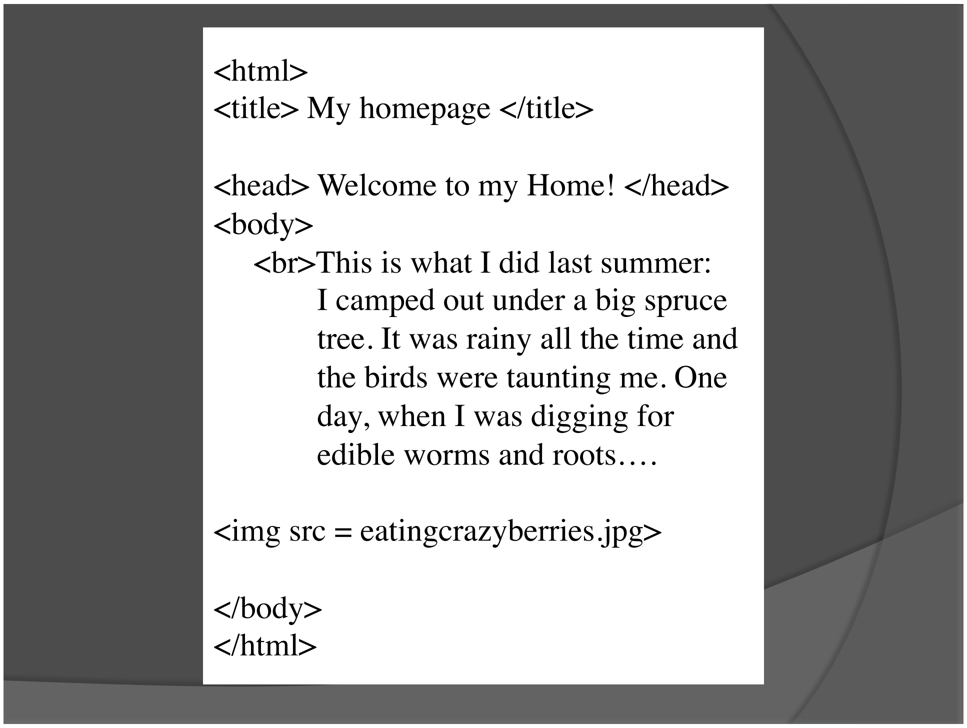
```
<html>
<title> My homepage </title>

<head> Welcome to my Home! </head>






</html>
```



```
<html>
<title> My homepage </title>

<head> Welcome to my Home! </head>
<body>









</body>
</html>
```

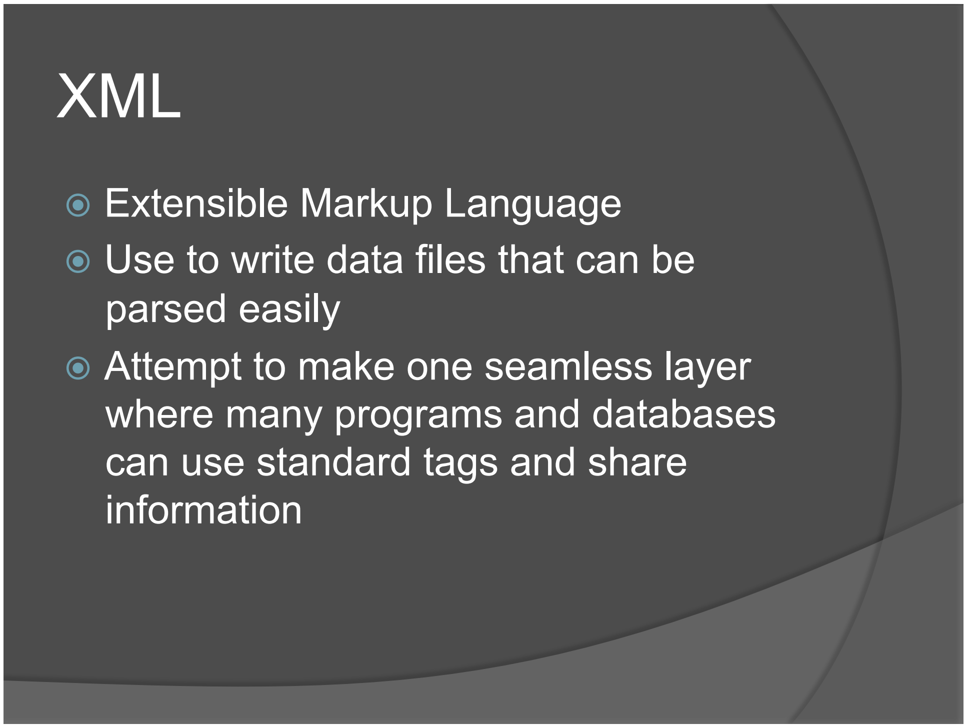


```
<html>
<title> My homepage </title>

<head> Welcome to my Home! </head>
<body>
   <br>This is what I did last summer:
        I camped out under a big spruce
        tree. It was rainy all the time and
        the birds were taunting me. One
        day, when I was digging for
        edible worms and roots….



</body>
</html>
```



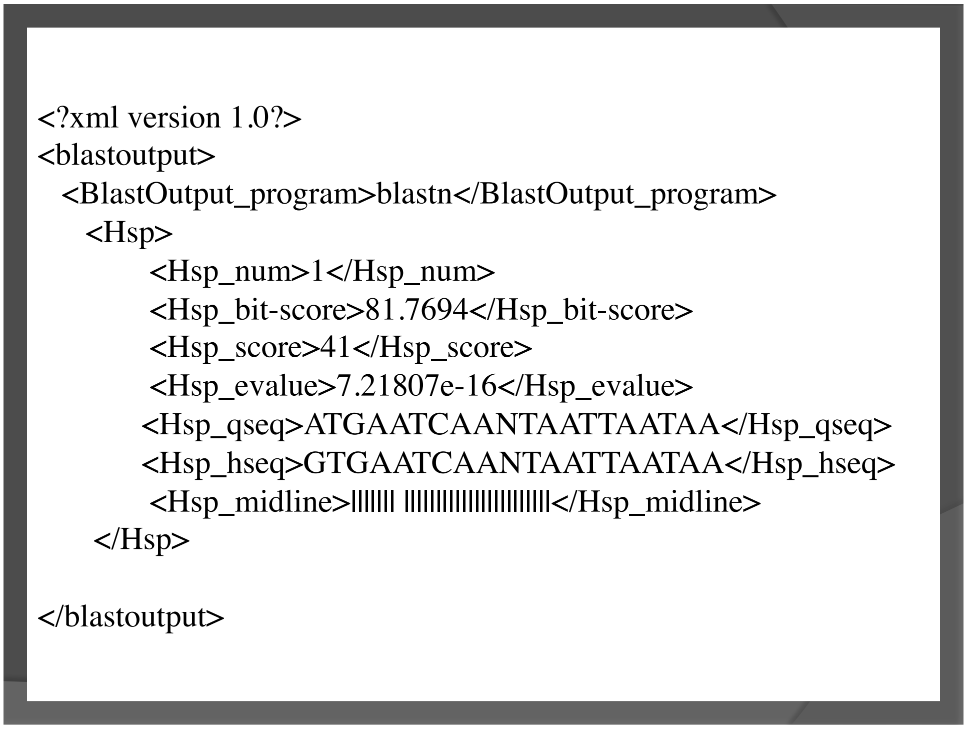
```
<html>
<title> My homepage </title>

<head> Welcome to my Home! </head>
<body>
    <br>This is what I did last summer:
          I camped out under a big spruce
          tree. It was rainy all the time and
          the birds were taunting me. One
          day, when I was digging for
          edible worms and roots….

<img src = eatingcrazyberries.jpg>

</body>
</html>
```

XML

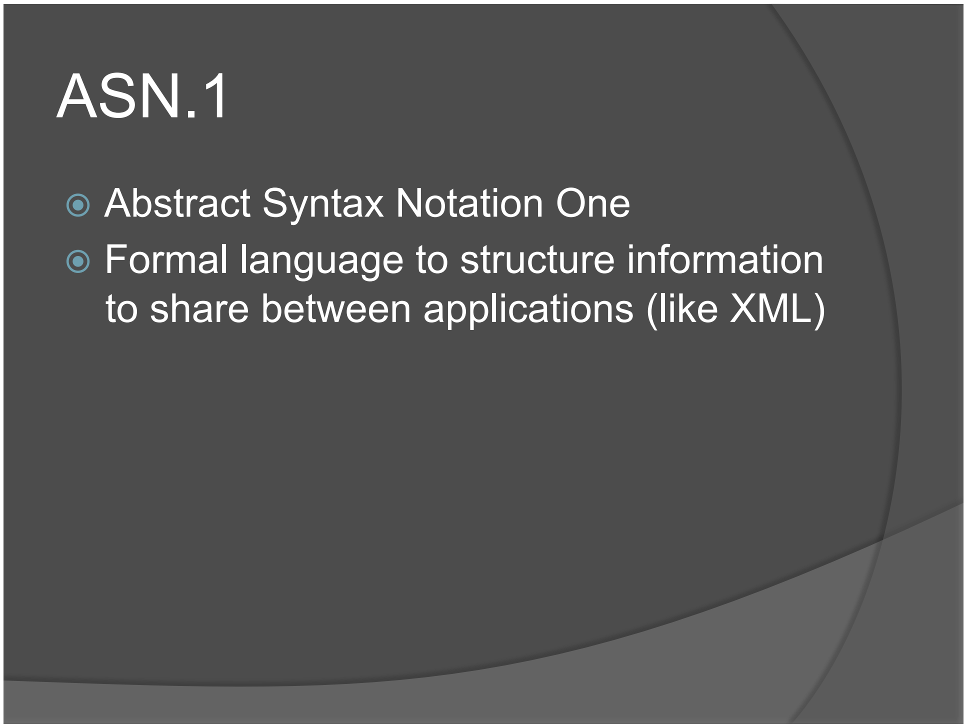


XML

- Extensible Markup Language
- Use to write data files that can be parsed easily
- Attempt to make one seamless layer where many programs and databases can use standard tags and share information

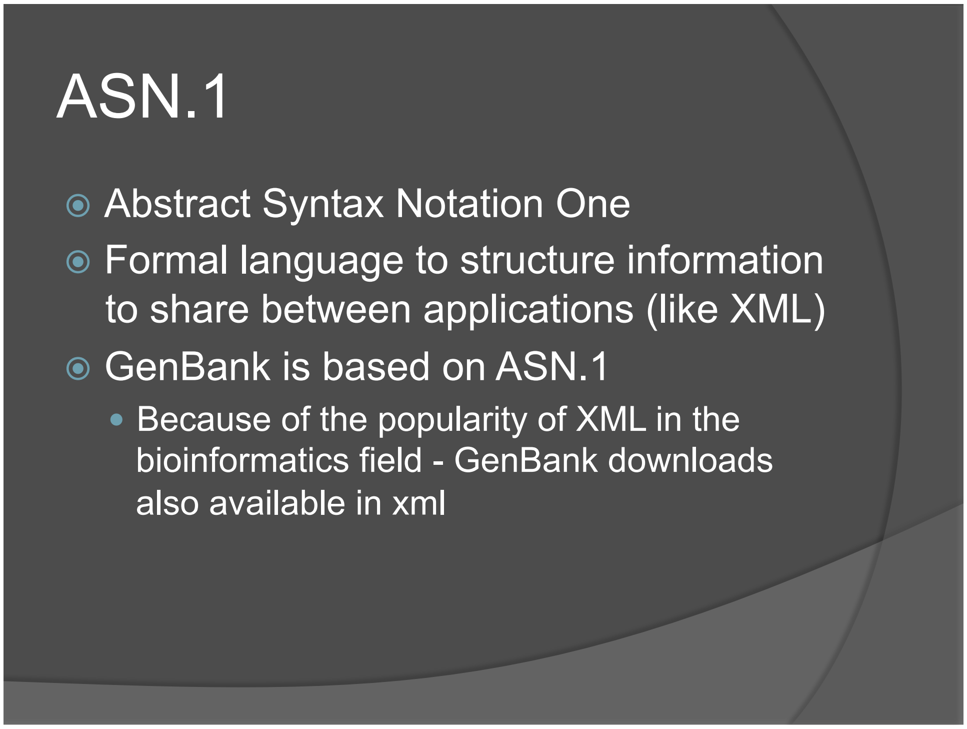


```
<?xml version 1.0?>
<blastoutput>
  <BlastOutput_program>blastn</BlastOutput_program>
    <Hsp>
          <Hsp_num>1</Hsp_num>
          <Hsp_bit-score>81.7694</Hsp_bit-score>
          <Hsp_score>41</Hsp_score>
          <Hsp_evalue>7.21807e-16</Hsp_evalue>
         <Hsp_qseq>ATGAATCAANTAATTAATAA</Hsp_qseq>
         <Hsp_hseq>GTGAATCAANTAATTAATAA</Hsp_hseq>
          <Hsp_midline>||||||| |||||||||||||||||||||||</Hsp_midline>
     </Hsp>

</blastoutput>
```



ASN.1

- Abstract Syntax Notation One
- Formal language to structure information to share between applications (like XML)



ASN.1

- Abstract Syntax Notation One
- Formal language to structure information to share between applications (like XML)
- GenBank is based on ASN.1
  - Because of the popularity of XML in the bioinformatics field - GenBank downloads also available in xml