Unix/Linux Tutorial for Beginners
Session II

Mihaela Martis

NBIS & Faculty of Medicine and Health Sciences
Division Cell Biology, IKE

NBIS NATIONAL BIOINFORMATICS INFRASTRUCTURE SWEDEN

elixir

LINKÖPING UNIVERSITY

The filesystem

- in UNIX everything is a file or a process

The filesystem

- in UNIX everything is a file or a process
  - *process* $\rightarrow$ is an executing program

The filesystem

- in UNIX everything is a file or a process
  - *process* $\rightarrow$ is an executing program
  - *file* $\rightarrow$ is a collection of data created by users using text editors, running programs
    $\rightarrow$ can be grouped into directories

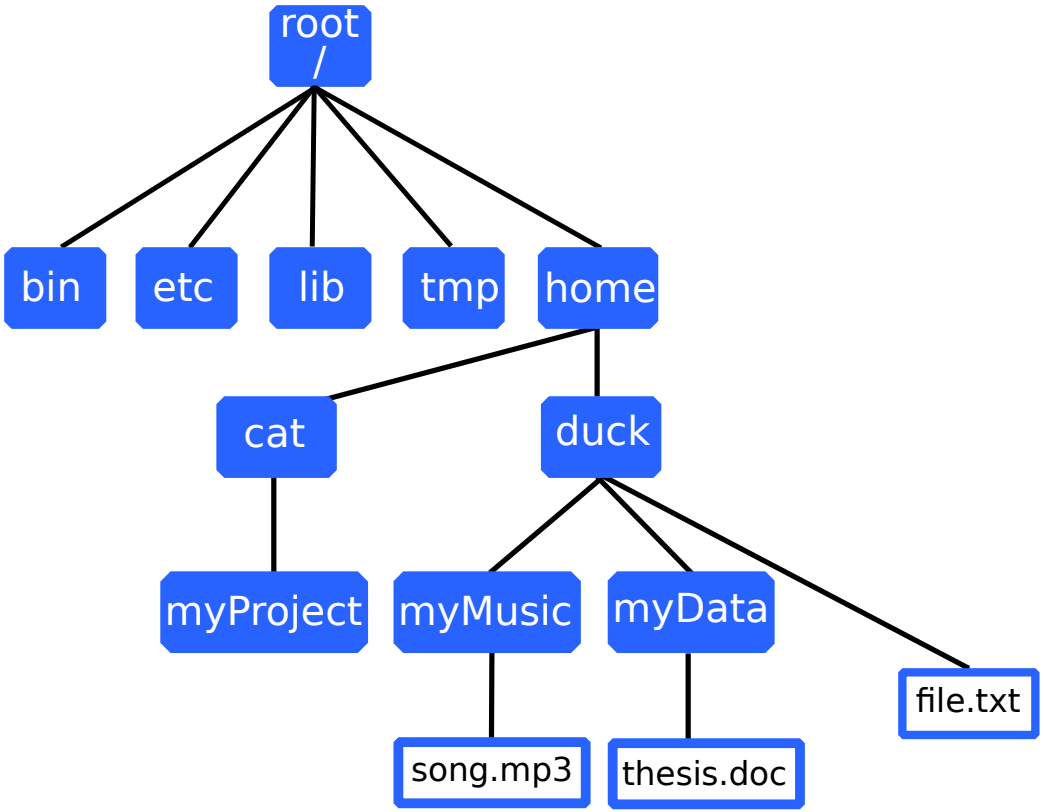
The filesystem

- in UNIX everything is a file or a process
  - *process* $\rightarrow$ is an executing program
  - *file* $\rightarrow$ is a collection of data created by users using text editors, running programs
    $\rightarrow$ can be grouped into directories
- a filesystem is a logical collection of files and directories on a disk managed using a hierarchical filesystem structure

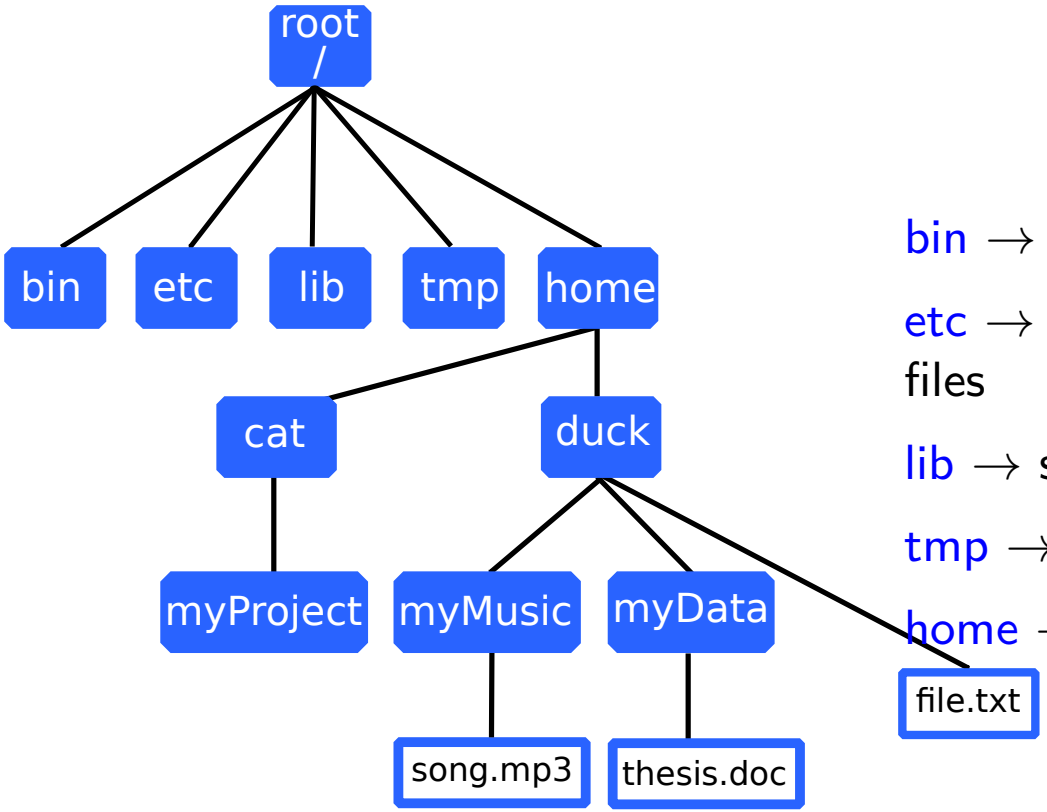
Filesystem properties

- represented as an upside-down tree with a root directory ('/') at the top
- each file or directory is uniquely identified by its name
- it is self contained $\rightarrow$ no dependencies between one filesystem and any other

Filesystem structure



Filesystem structure



bin → shared executable files

etc → administrative and configuration files

lib → shared library files

tmp → temporary files storage place

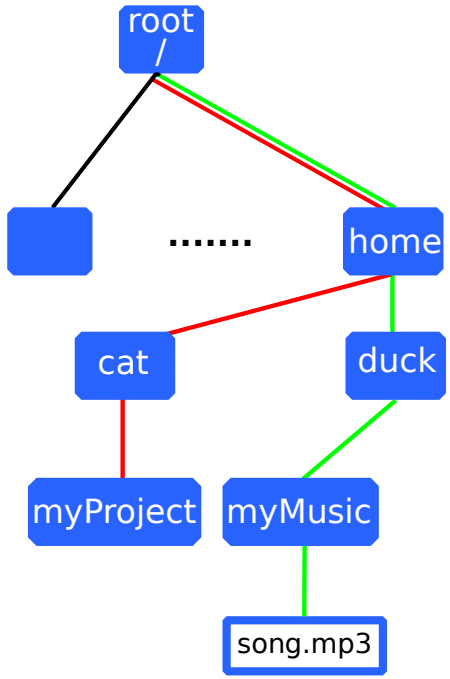
home → users home directories

Path

- unique location to a file or directory
- a combination of '/' and alpha-numeric characters

  /home/duck/myMusic/song.mp3

  /home/cat/myProject
- '/' has 2 meanings:
  - root → in front of a file/directory
  - a separator → inside a path

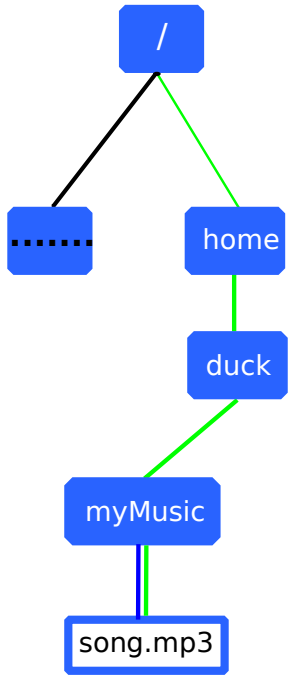


Absolute vs relative path

- absolute path → specify the location of a file or directory starting from the root '/'

  → /home/duck/myMusic/song.mp3
- relative path → defined as a path related to the present working directory

  → song.mp3



Current and parent directories

- each directory has 2 special subdirectories:
  - . → **current working directory** - is the directory you are in when running a command
  - .. → **parent directory** – refers to the directory containing the current directory
- **song.mp3**
  - current working directory: */home/duck/myMusic* or *./song.mp3*
  - parent directory: */home/duck*

File names

- are case sensitive
- forbidden character: '/'
- not recommended: [space] [enter] ; * [quotes]
- have usually the form: 'name.something'

File names

- are case sensitive
- forbidden character: '/'
- not recommended: [space] [enter] ; * [quotes]
- have usually the form: 'name.something'
- '.something'= filename extension
- some extensions are used by convention to indicate the type of data the file holds

File names

- are case sensitive
- forbidden character: '/'
- not recommended: [space] [enter] ; * [quotes]
- have usually the form: 'name.something'
- '.something'= filename extension
- some extensions are used by convention to indicate the type of data the file holds
  - .sh, .pl, .py → shell, perl, and python scripts
  - .txt → plain text file
  - .pdf → PDF document
  - .doc → MS Word document
  - .csv → plain text file, tabular data, separated by commas
  - .tab → plain text file, tabular data, separated by tab

File type

- file → reports the type of a file

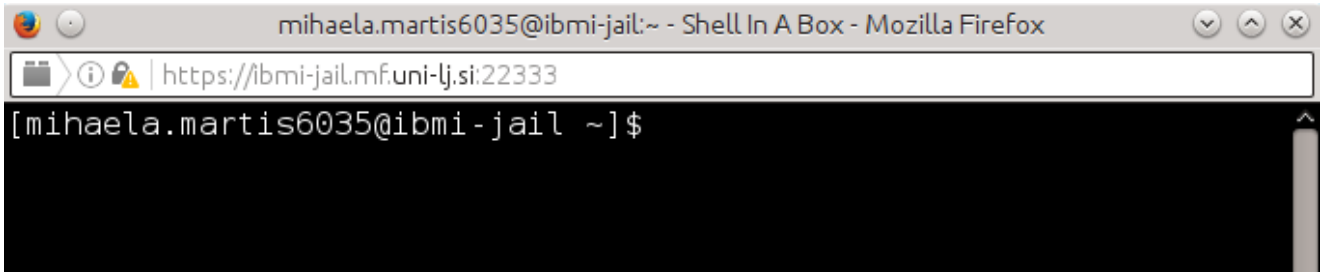
```
$ file mySeq.fa
mySeq.fa : ASCII text

$ file unix_introduction.tex
unix_introduction.tex: LaTeX 2e document, UTF-8 Unicode
    text

$ file exercises.pdf
exercises.pdf: PDF document, version 1.5
```

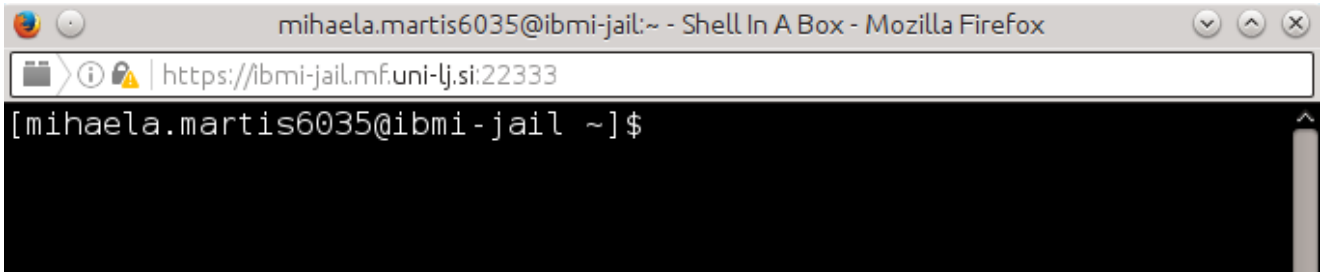
Navigating the filesystem

- 'home' directory → default directory for a specific user
- it's represented on the prompt by the sign '~'



Navigating the filesystem

- 'home' directory → default directory for a specific user
- it's represented on the prompt by the sign '~'

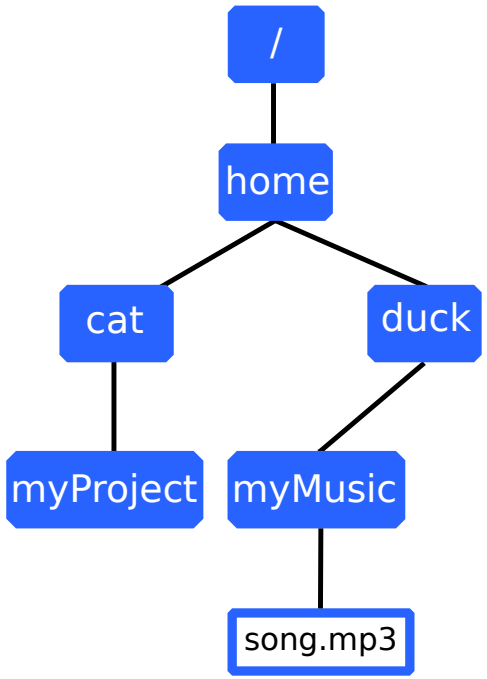


- pwd → 'print working directory' – writes the full pathname of the current directory

cd – change directory

- it changes the shell's idea of what directory we are in

```
$ cd myMusic
$ pwd
/home/duck/myMusic
```



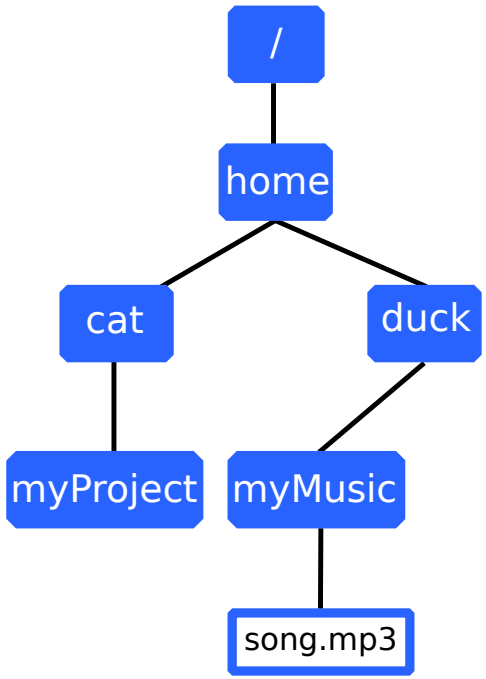
cd – change directory

- it changes the shell's idea of what directory we are in

```
$ cd myMusic
$ pwd
/home/duck/myMusic
```

- how to go up/back using absolute paths?

```
$ cd /home/cat/myProject
```



cd – change directory

- it changes the shell's idea of what directory we are in

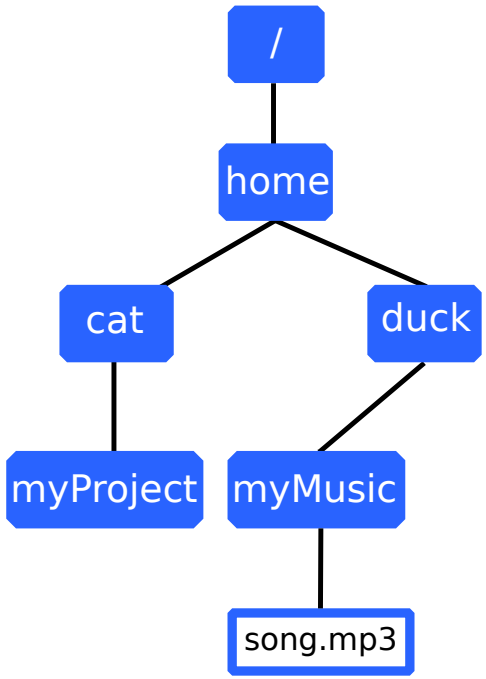
```
$ cd myMusic
$ pwd
/home/duck/myMusic
```

- how to go up/back using absolute paths?

```
$ cd /home/cat/myProject
```

- how to go up/back using relative paths?

```
$ cd ../../cat/myProject
```



Tilde Expansion

- tilde character ('~') has a special meaning
- used at the beginning of a word, it expands into the name of the home directory:

  /home/duck is equivalent with ~

```
$ pwd
/home/duck/myMusic/adele
$ cd ~
$ pwd
/home/duck
```

ls – list the content of directories

- ls → 'lists' in alphabetical order the file and directory names of the current directory

```
$ pwd
/home/duck/myMusic
$ ls
adele  coldplay.mp3  song.mp3
```

ls – list the content of directories

- ls → 'lists' in alphabetical order the file and directory names of the current directory

```
$ pwd
/home/duck/myMusic
$ ls
adele  coldplay.mp3  song.mp3
```

- many options available, e. g. :
  - **-F**, adds a trailing '/' to the names of directories
  - **-l**, uses a long listing format, including the access rights
  - **-h**, displays the file & directory sizes in human readable format
  - **-r**, reverses the order while sorting
  - **-a**, 'show all', forces ls to show us all files and directories

ls – examples

```
$ pwd
/home/duck/myMuic
$ ls -F
adele/  coldplay.mp3  song.mp3
$ ls -a
.  ..  adele  coldplay.mp3  song.mp3
$ ls -r
song.mp3  coldplay.mp3  adele

$ ls -l
total 8
drwxrwxr-x 2 duck duck   6 Oct 10 11:34 adele
-rw-rw-r-- 1 duck duck  29 Oct 10 11:34 coldplay.mp3
-rw-rw-r-- 1 duck duck 119 Oct 10 11:34 song.mp3

$ ls -hl
total 8.0K
drwxrwxr-x 2 duck duck   6K Oct 10 11:34 adele
-rw-rw-r-- 1 duck duck  29K Oct 10 11:36 coldplay.mp3
-rw-rw-r-- 1 duck duck 119K Oct 10 11:37 song.mp3
```

Hidden files

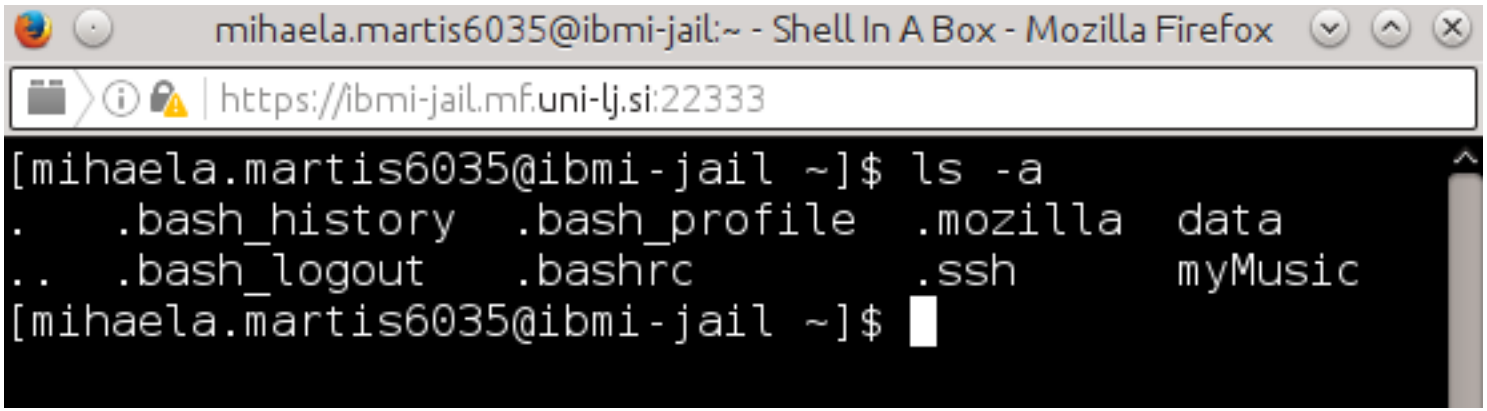
- in Unix/Linux hidden files and directories are usually configuration files
- use **ls -a** to display them

Hidden files

- in Unix/Linux hidden files and directories are usually configuration files
- use **ls -a** to display them
- e. g. : the current '.' and parent '..' directories

Hidden files

- in Unix/Linux hidden files and directories are usually configuration files
- use **ls -a** to display them
- e. g. : the current '.' and parent '..' directories



Wildcards

- are symbols used to replace or represent one or more characters
- are often used with file or directory names and commands
- basic wildcards:
  - * – matches zero or more characters
  - ? – matches exactly one character
  - [] – matches a range of characters (0-9, a-Z)
  - – specify a list of terms separated by commas
  - \ – used as an 'escape' character

Wildcard examples I

- list all files ending with '.mp3' or starting with 'c'

```
$ ls *.mp3
coldplay.mp3  song.mp3

$ ls c*
coldplay.mp3
```

- list all files named 'song', followed by single character and ending with '.mp3'

```
$ ls
adele  coldplay.mp3  song.mp3  song1.mp3  song2.mp3  song3.mp3  song4.mp3

$ ls song?.mp3
song1.mp3  song2.mp3  song3.mp3  song4.mp3
```

Wildcard examples II

- list all files with the numbers 1 or 2 in their names

```
$ ls *[1-2]*
song1.mp3  song2.mp3
```

- list all files starting with a 'c' and those containing the numbers 2 to 4 in their name

```
$ ls {c*,*[2-4].mp3}
coldplay.mp3  song2.mp3  song3.mp3  song4.mp3

$ ls {c*,*[2:4].mp3}
coldplay.mp3  song2.mp3  song4.mp3
```

Tree

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

```
$ pwd
/home/duck/myMusic
$ tree
.
|-- adele
|-- coldplay.mp3
|-- song.mp3
|-- song1.mp3
|-- song2.mp3
|-- song3.mp3
`-- song4.mp3
1 directory, 6 files
```

Tree

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

```
$ pwd
/home/duck/myMusic
$ tree
.
|-- adele
|-- coldplay.mp3
|-- song.mp3
|-- song1.mp3
|-- song2.mp3
|-- song3.mp3
`-- song4.mp3
1 directory, 6 files
```

- tree -d – list directories only

```
$ tree -d
.
`-- adele
1 directory.
```

How to create a directory?

- mkdir (make directory) – to create new directories
- usage: mkdir [OPTION] NAME

```
$ pwd
/home/duck
$ ls -F
data/  myMusic/
$ mkdir myPublications
$ ls -F
data/  myMusic/  myPublications/
```

- if directory already exists, it reports an error

```
$ mkdir data
mkdir: cannot create directory 'data': File exists
```

mkdir -p option

- -p – no error if existing, make parent directory if needed

```
$ ls -F
data/  myMusic/  myPublications/
$ mkdir -p myPublications
$ ls -F
data/ myMusic/  myPublications/

$ cd myPublications
$ mkdir thesis/references
mkdir: cannot create directory 'thesis/references': No
such file or directory

$ mkdir -p thesis/references
$ ls -F
thesis/
$ cd thesis
$ ls -F
references/
```

mv – moving files

- mv – moves files or directories from one location to another one
- usage: mv <file/directory> <destination>

```
$ ls
adele  coldplay.mp3  song.mp3
$ mv coldplay.mp3 /home/cat/myProject
$ ls
adele  song.mp3
$ cd /home/cat/myProject
$ ls
coldplay.mp3
```

mv – moving files

- mv – moves files or directories from one location to another one
- usage: mv <file/directory> <destination>

```
$ ls
adele  coldplay.mp3  song.mp3
$ mv coldplay.mp3 /home/cat/myProject
$ ls
adele  song.mp3
$ cd /home/cat/myProject
$ ls
coldplay.mp3
```

- mv – renames a file or a directory

```
$ ls
adele  song.mp3
$ mv adele adele_songs
$ ls
adele_songs  song.mp3
```

Copying files

- cp – copies a file instead of moving it
- cp <source folder> <destination folder>

```
$ pwd
/home/duck/myMusic
$ cp coldplay.mp3 coldplay_upAndUp.mp3
$ ls
coldplay.mp3 coldplay_upAndUp.mp3

$ cp coldplay.mp3 ../coldplay_backup.mp3
$ cd ..
$ ls
coldplay.mp3  data  myMusic  myPublications

$ cp -r adele_songs /home/duck/backup
$ ls /home/duck/backup
adele_songs
```

Removing files and directories

- rm – removes files and directories

```
$ rm draft.txt
```

- -i – asks for every deletion to be confirmed

```
$ rm -i draft.txt
rm: remove regular file 'draft.txt'? yes
```

- -r – removes everything in the directory, then the directory itself

```
$ rm thesis/references
rm: cannot remove 'thesis/references/': Is a
directory
$ rm -r thesis/references
```

- the shell **doesn't** have a trash bin → deleted files cannot be recovered

rmdir

- rmdir – removes an empty directory
- usage: rmdir <directory name>

```
$ cd ~/myPublications/thesis
$ ls -F
references/
$ rmdir references
$ ls
$
```

Summary

- **filesystem** $\rightarrow$ controls how data is stored and retrieved
- **absolute path** $\rightarrow$ specifies the location of a file or directory from the root directory (/)
- **relative path** $\rightarrow$ path related to the present working directory

Summary

- **filesystem** → controls how data is stored and retrieved
- **absolute path** → specifies the location of a file or directory from the root directory (/)
- **relative path** → path related to the present working directory
- filesystem commands:
  - **pwd** – displays the path of the current working directory
  - **cd** – change directory
  - **ls** – lists the content of directories
  - **mv** – move/rename a file or directory
  - **cp** – copies files and directories
  - **mkdir** – create new directories
  - **rm** – removes files and directories
  - **rmdir** – remove empty directories
  - **tree** – list contents of directories in a tree-like format.
  - **file** – returns the type of a file