



Introduction to Linux for Bioinformatics

The structure of Linux

Joachim Jacob
5 and 12 May 2014



This presentation is available under the Creative Commons Attribution-ShareAlike 3.0 Unported License. Please refer to <http://www.bits.vib.be/> if you use this presentation or parts hereof.



Meet your Linux system

We'll explore how a Linux system is organised

- into drives
- into partitions
- into folders
- into files

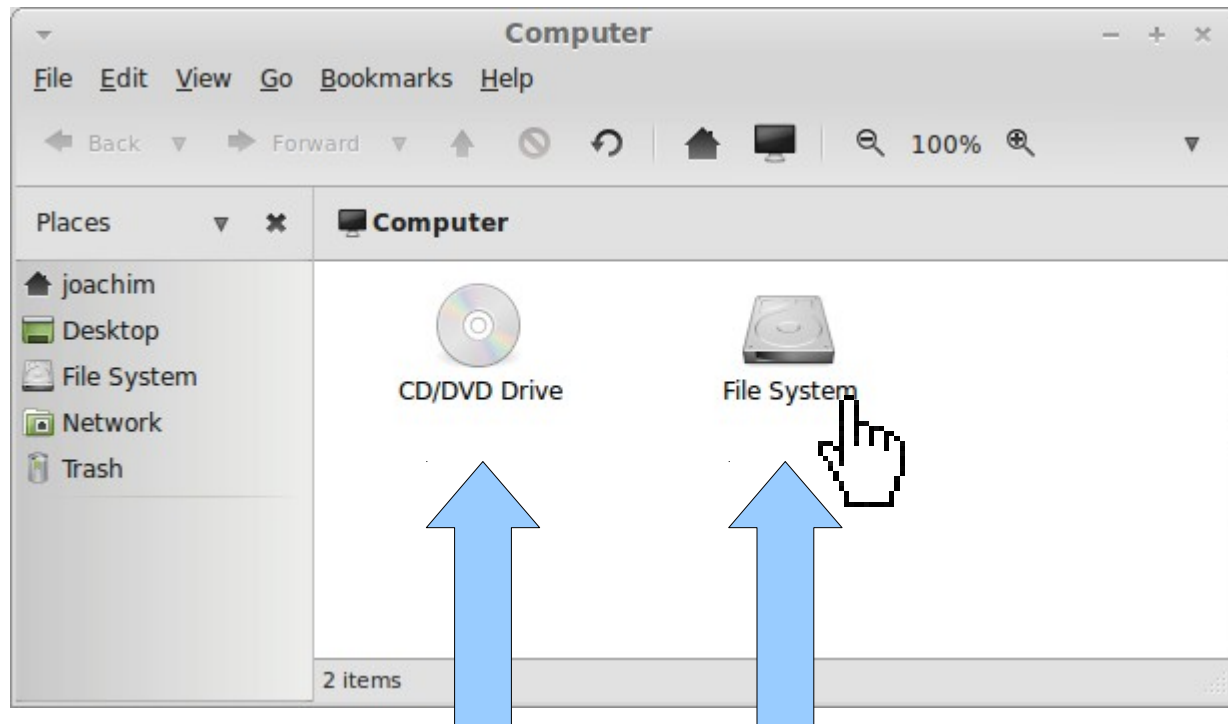


One of the most basic tasks in bioinformatics is handling and storing of big data.

Meet your Linux system

Follow a along: open  on the desktop.

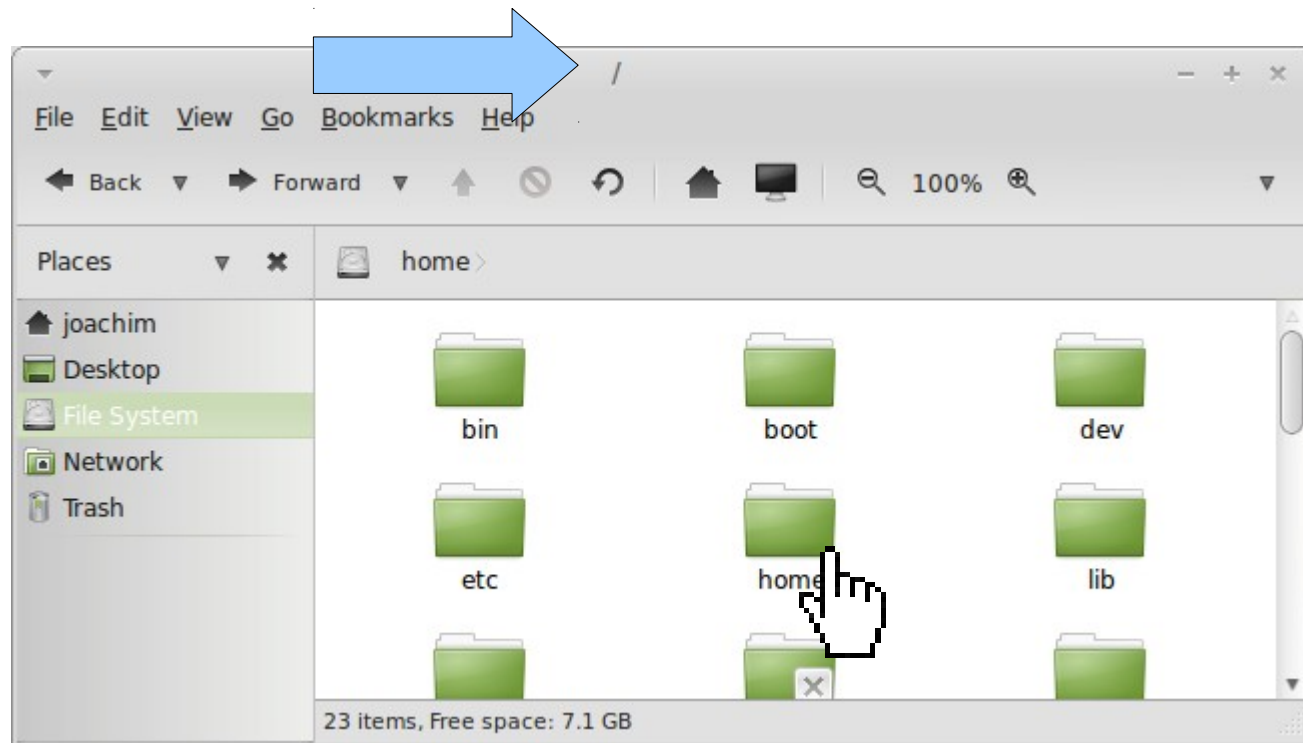
('Computer' is specific for Linux Mint, won't find it on other Linuxes)



The root directory

The disk on your computer, which has Linux installed on, is called **/** or the **root** directory. It is the start of the file system.

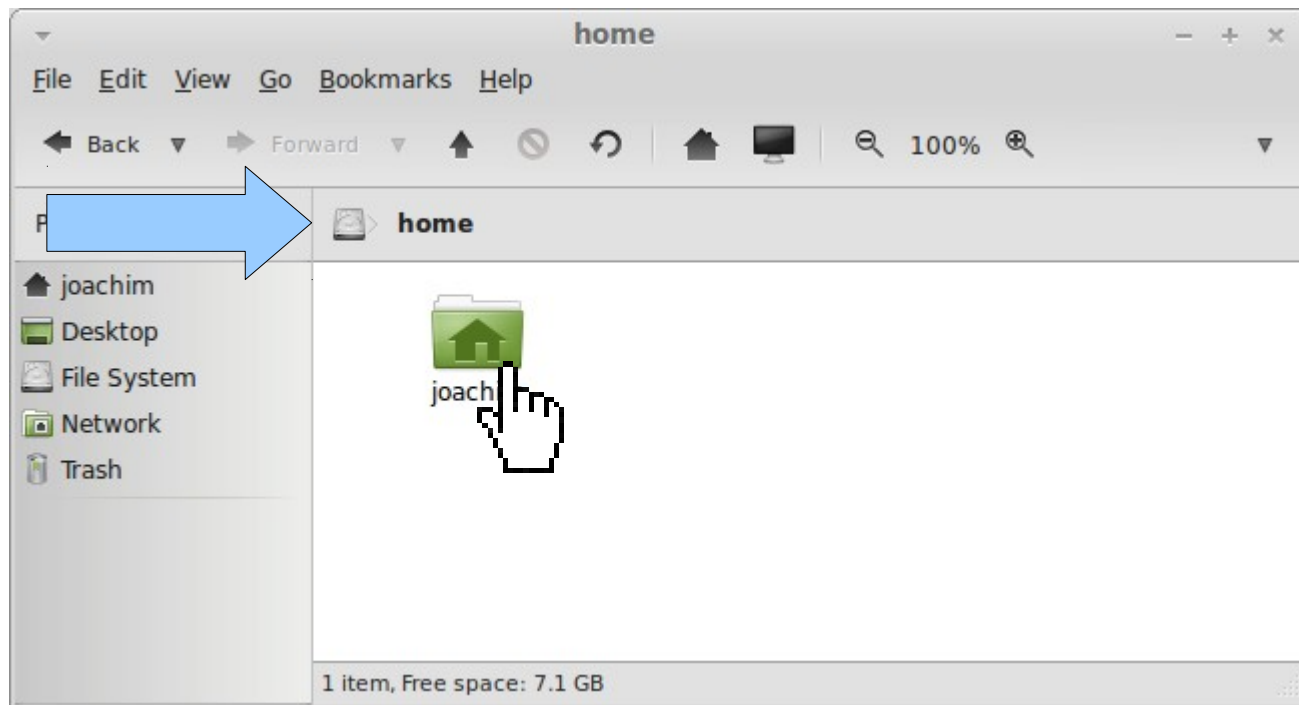
Now enter the folder home.



The home folder

One of the folders under the root directory is called **home**.

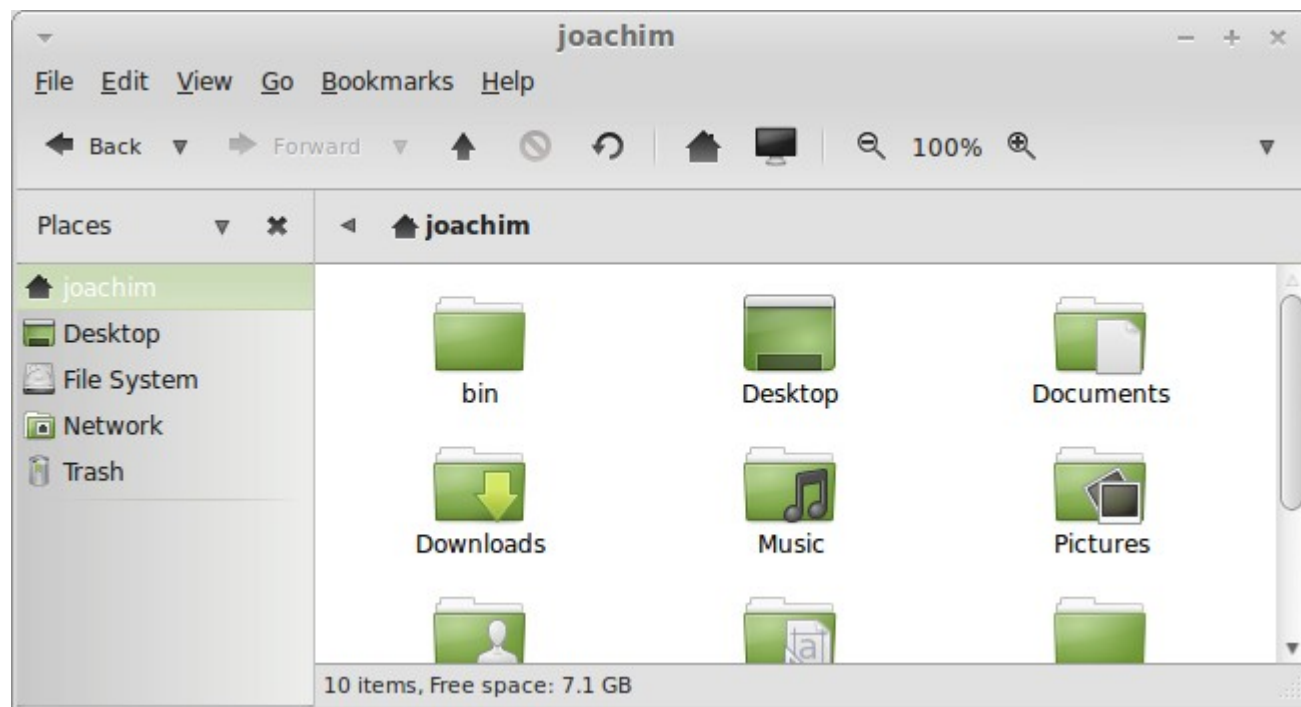
Every user on the system (server or PC) has one directory in /home. Locate yours and double-click on it.



Permissions in your home

Linux is very secure. **Only** in **your own home** folder you can **create** files and folders, usually not anywhere else.

Anywhere else is reserved for the administrator ('root')

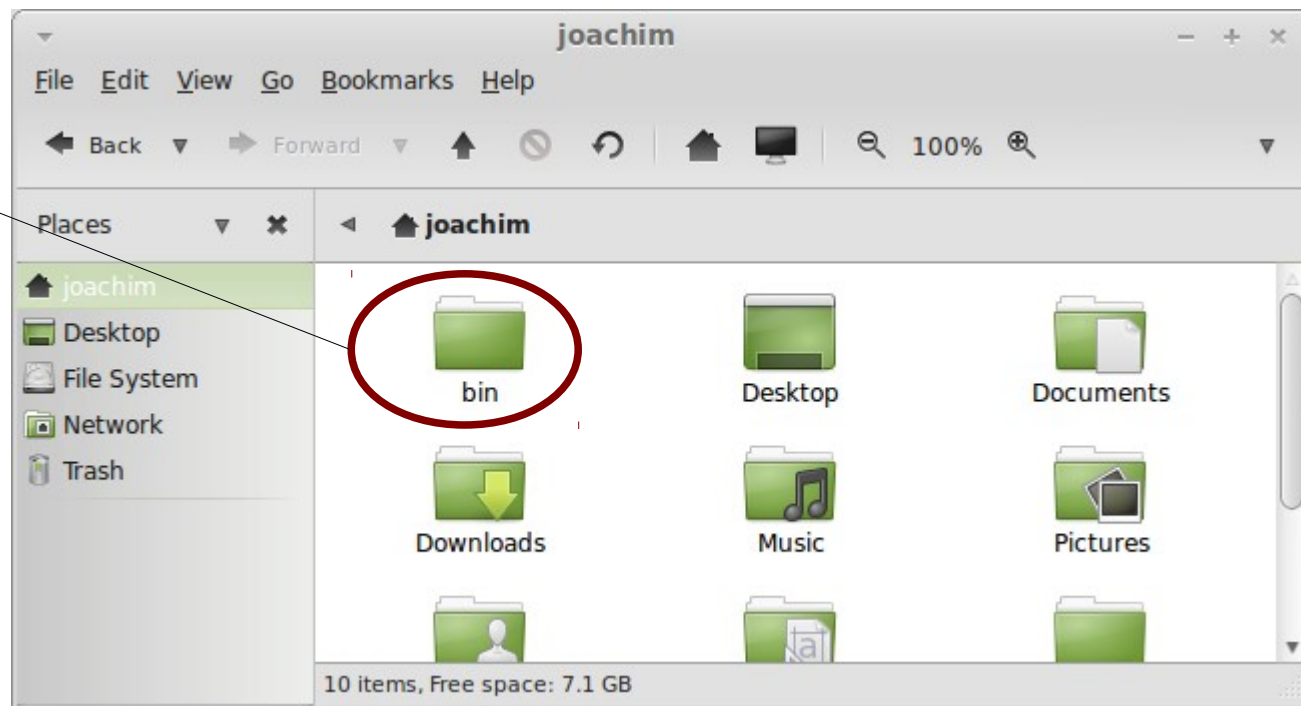


Create the folder bin in your home

Use the mouse (right-click), or press ctrl+shift+n, or, go via the File menu.

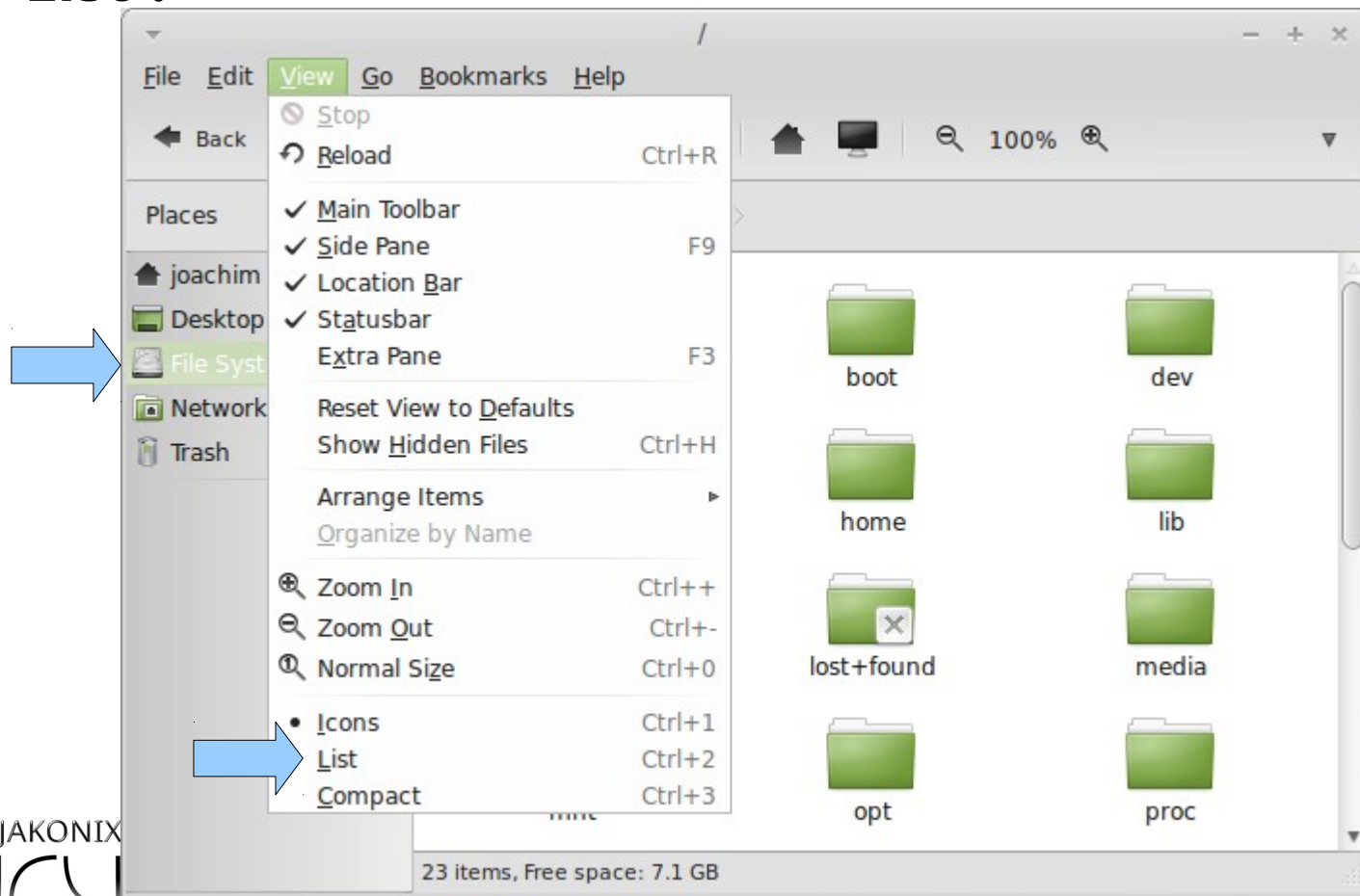
Create the folder bin. Later, we will put scripts in this folder.

Create
this
folder



Visualize the tree structure

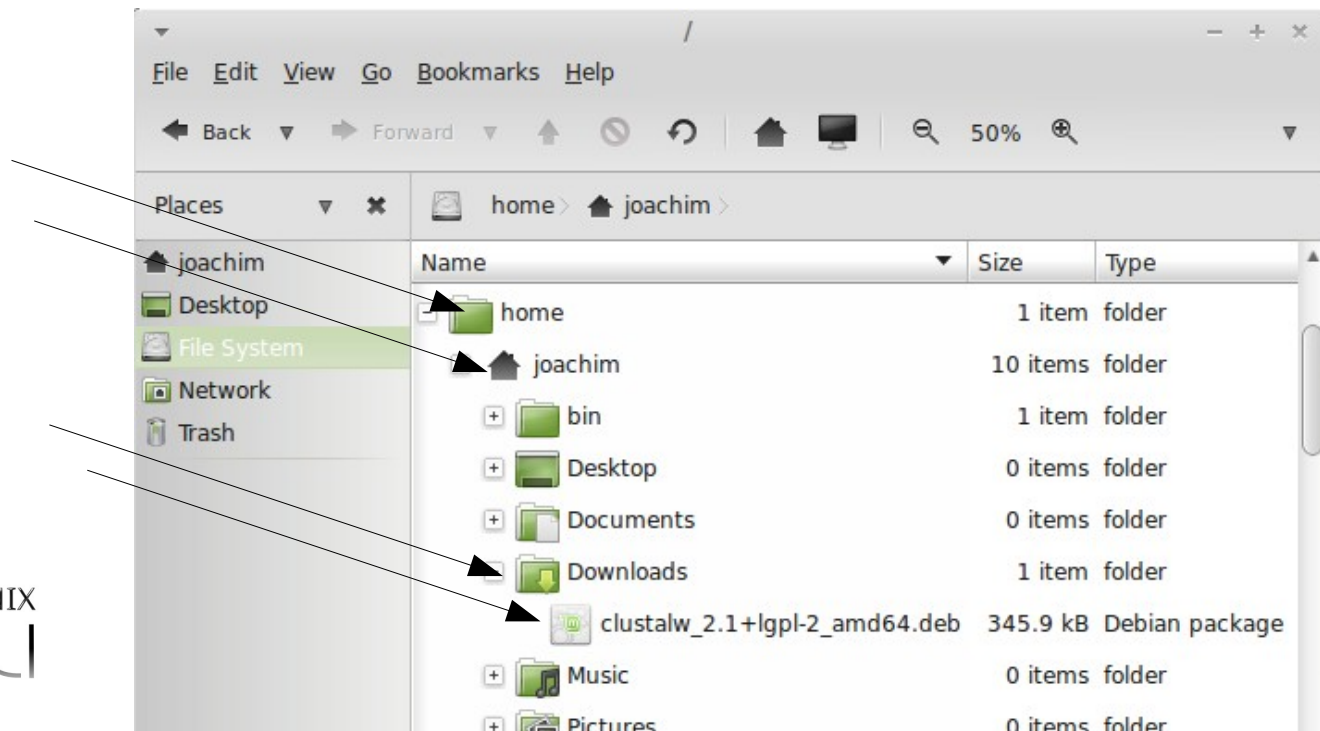
Go back to the root directory. Change the **view to 'List'**.



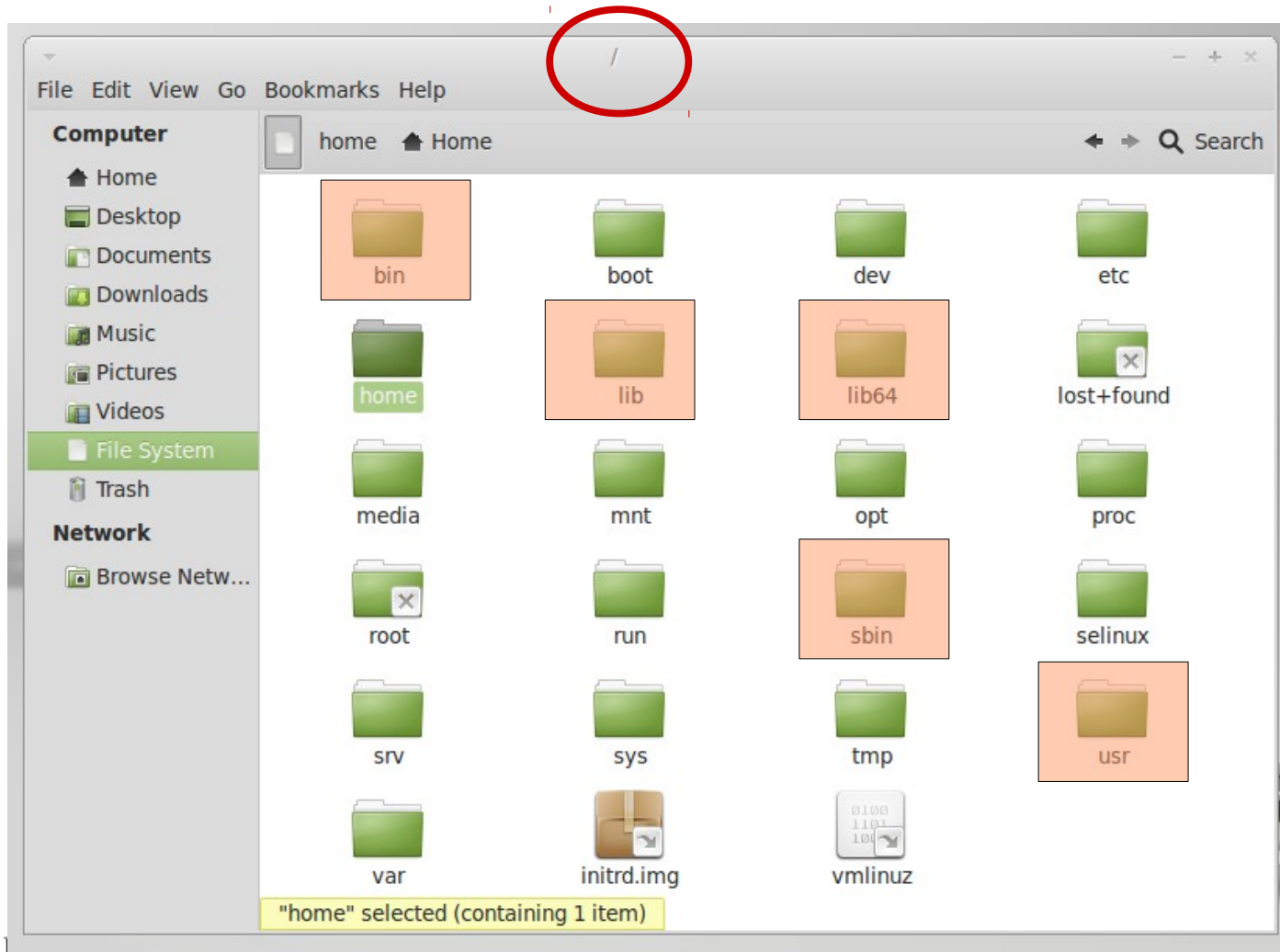
Visualize the tree structure

Clicking on the '+' expands the contents of that folder. You can do this for every subfolder too. A **path points to a location of a file or folder**. It is separated by a '/' symbol.

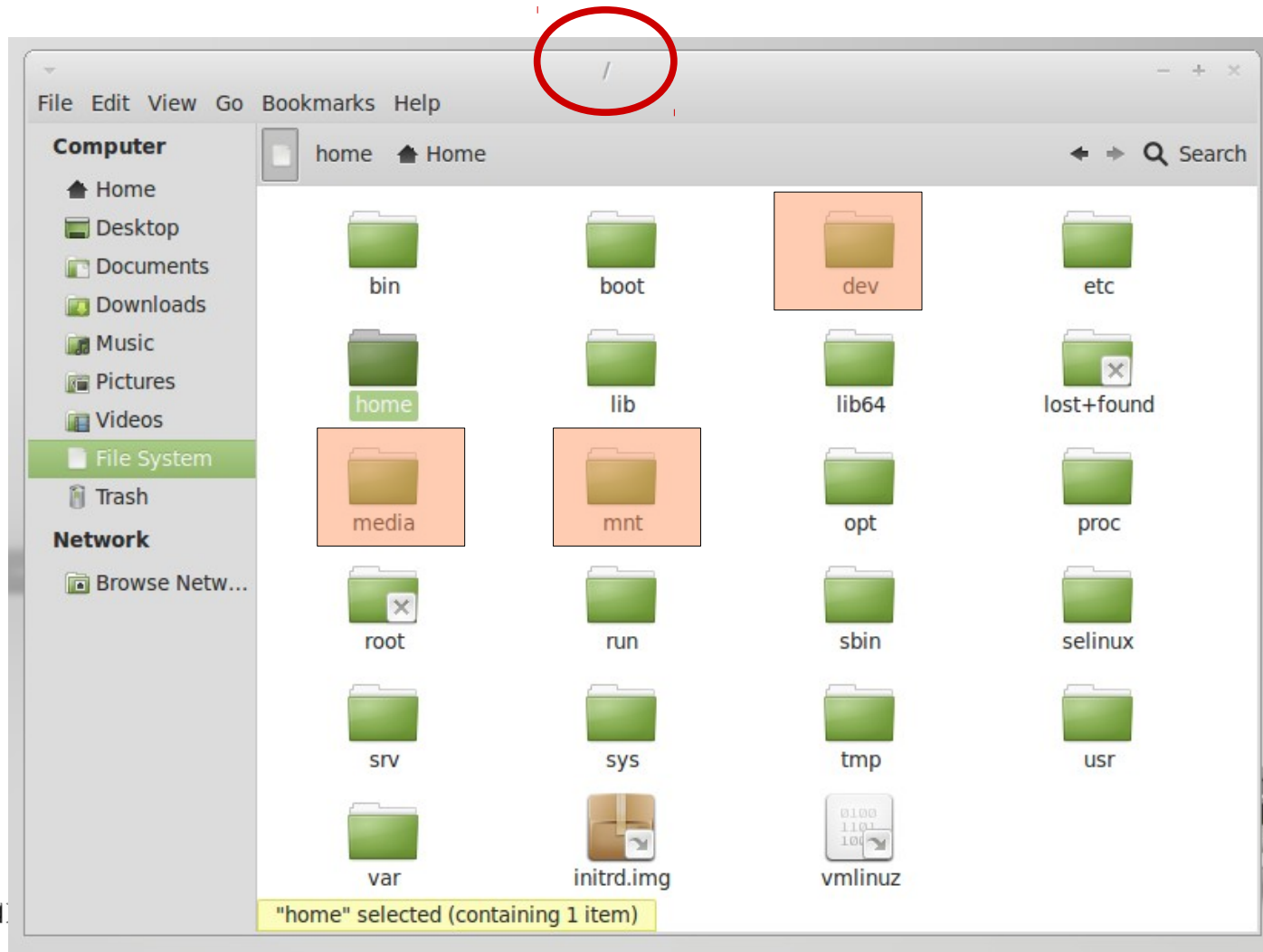
`/home/joachim/Downloads/clustalw_2.1+lgpl-2_amd64.deb`



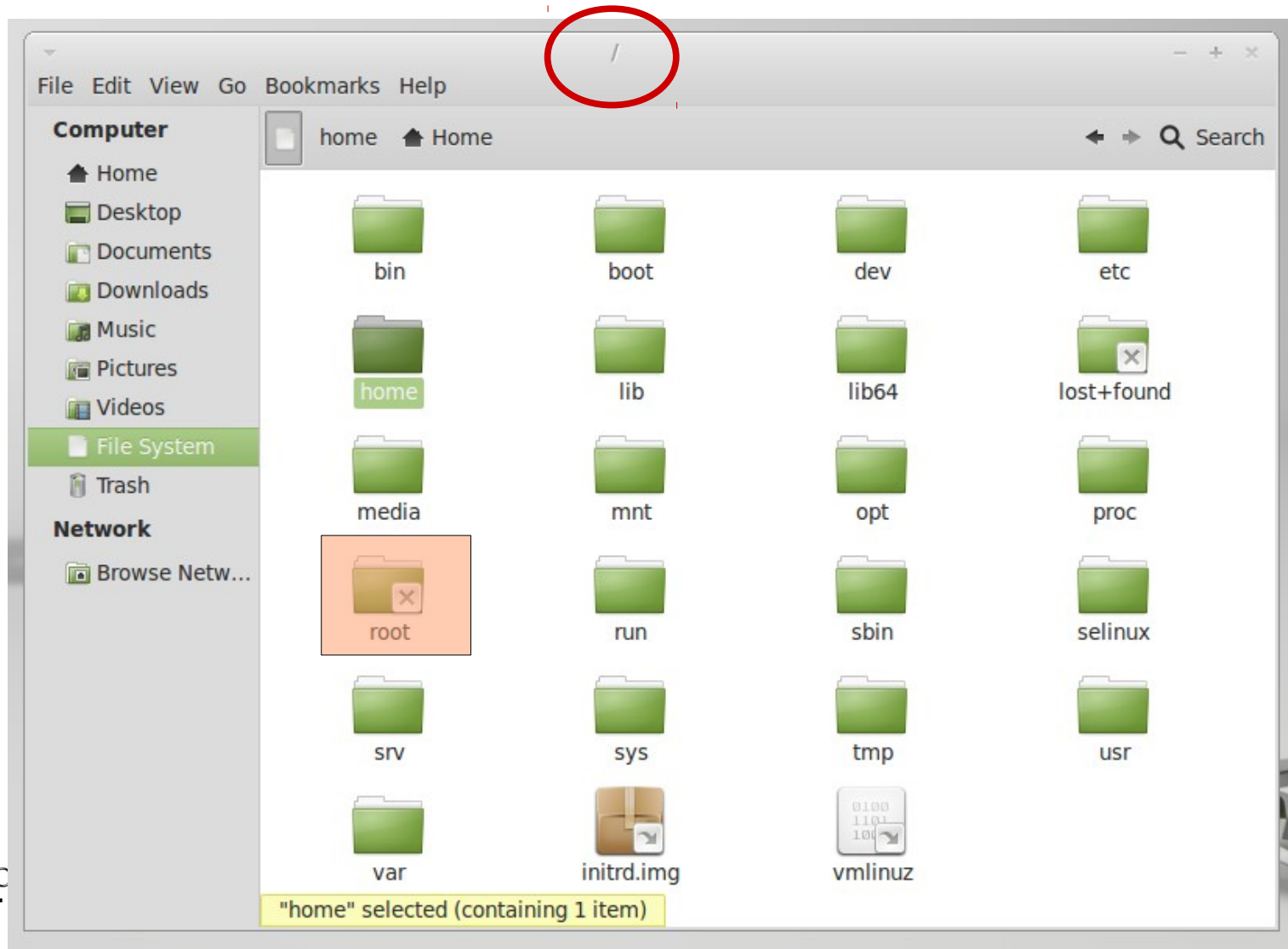
The program file's location



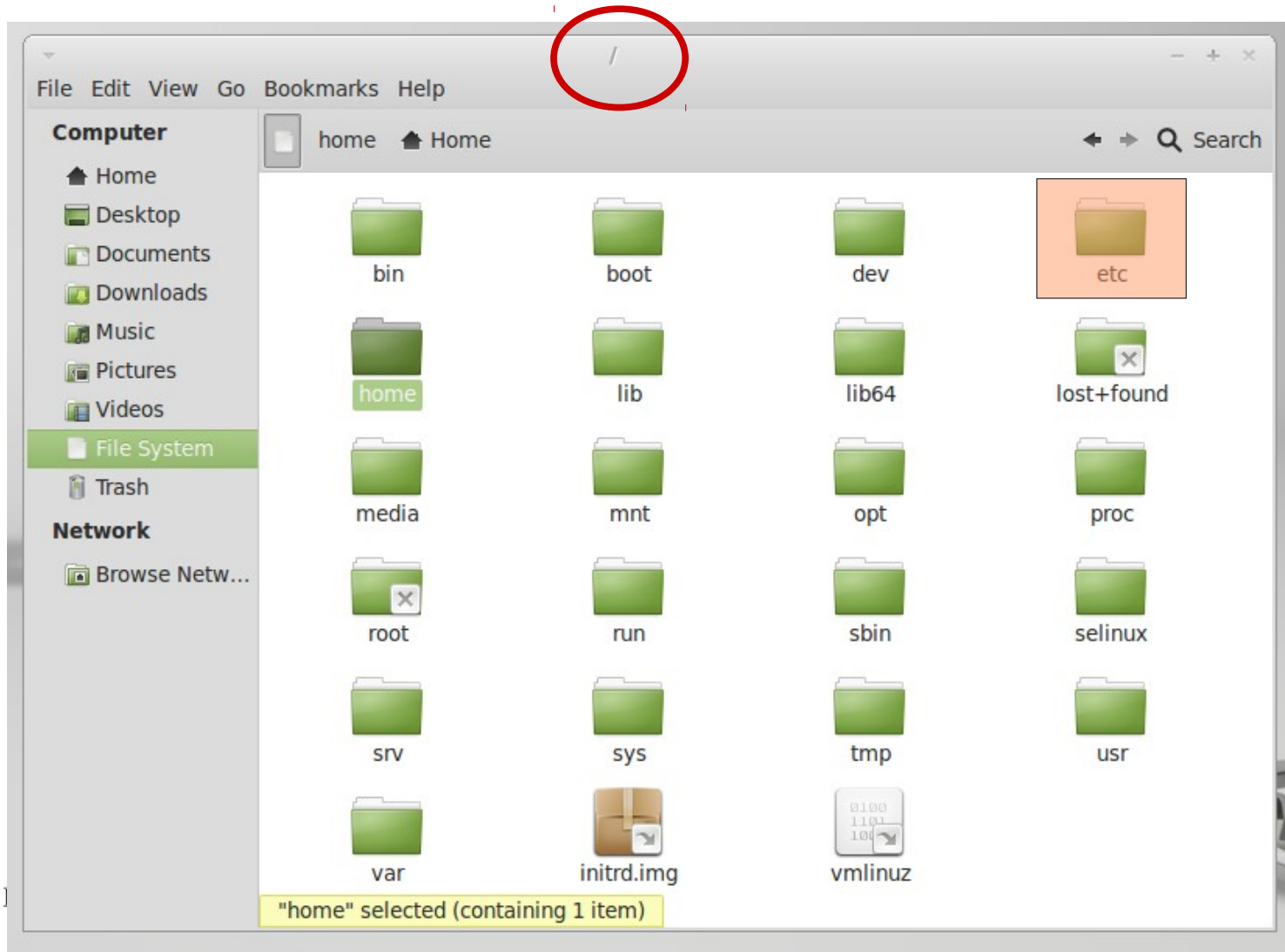
Disk and shares information



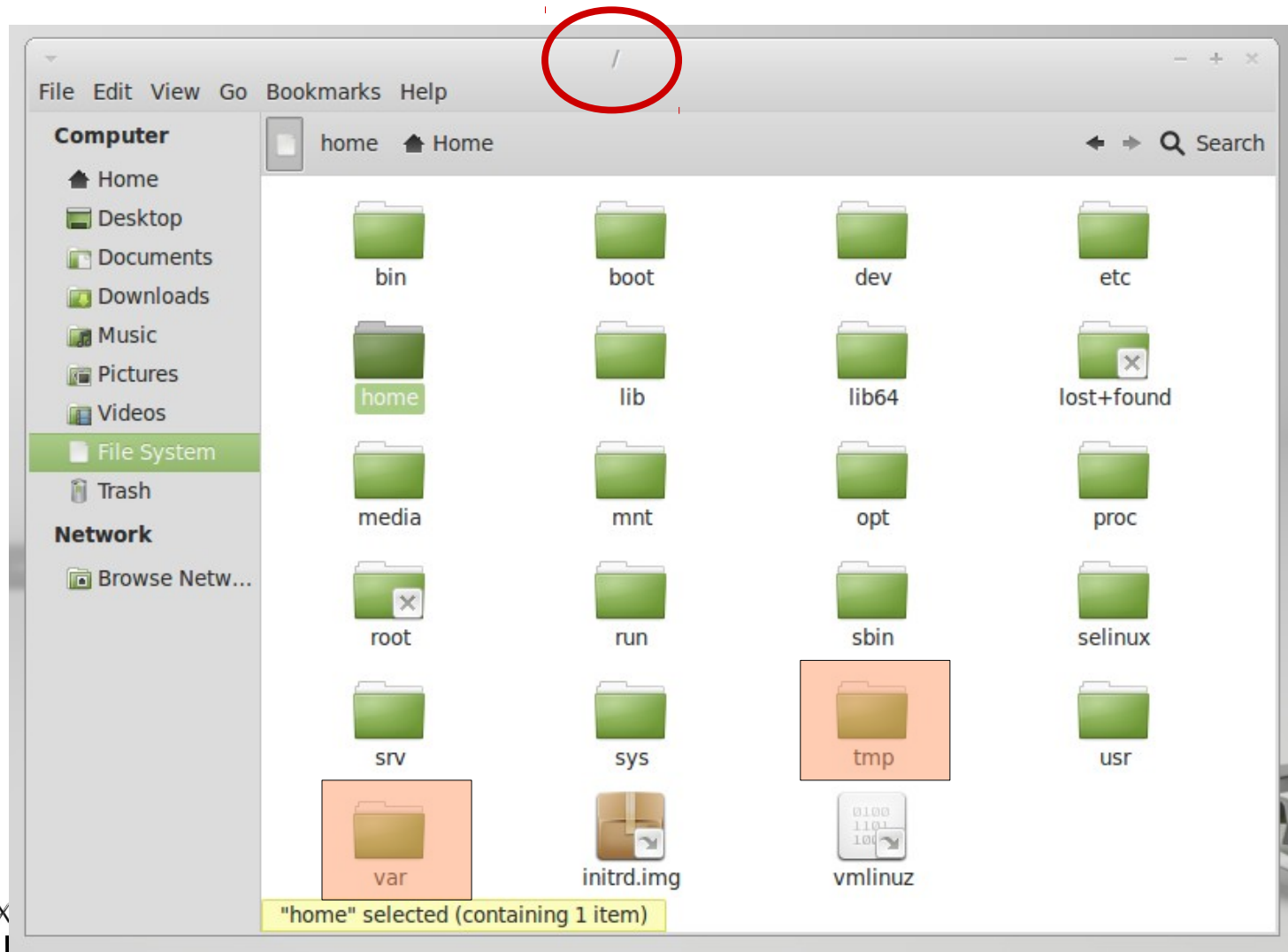
The administrator's home



Configuration files



Quickly changing content



'Everything is a file in Linux'



/dev/sda



/proc/meminfo



/dev/mouse1



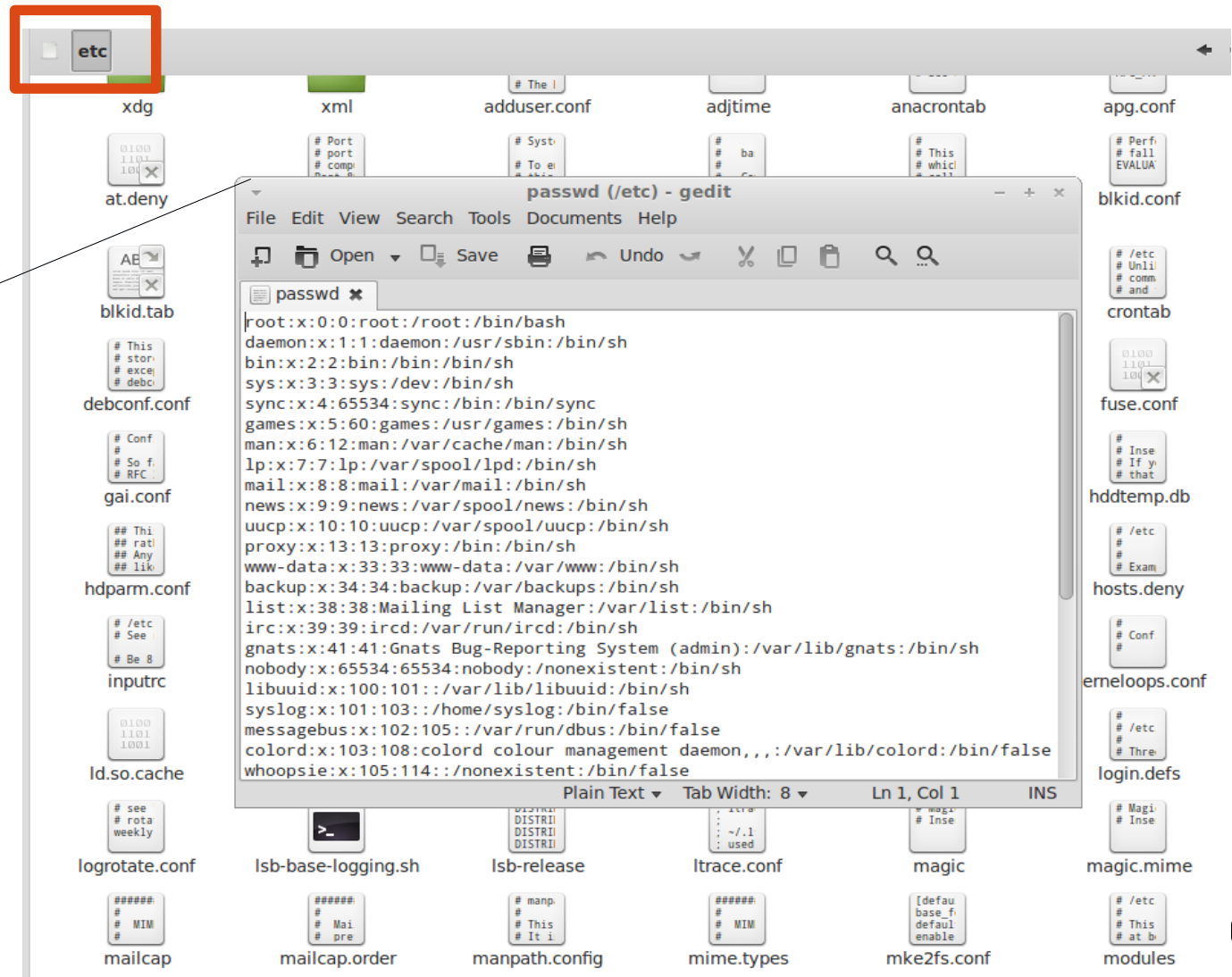
/dev/input1

“Everything is a **file** in linux”: **devices**, and their statuses are accessible by reading the contents of the paths to the respective files.

Note: but only *text files* can be displayed in human readable format in text editors.

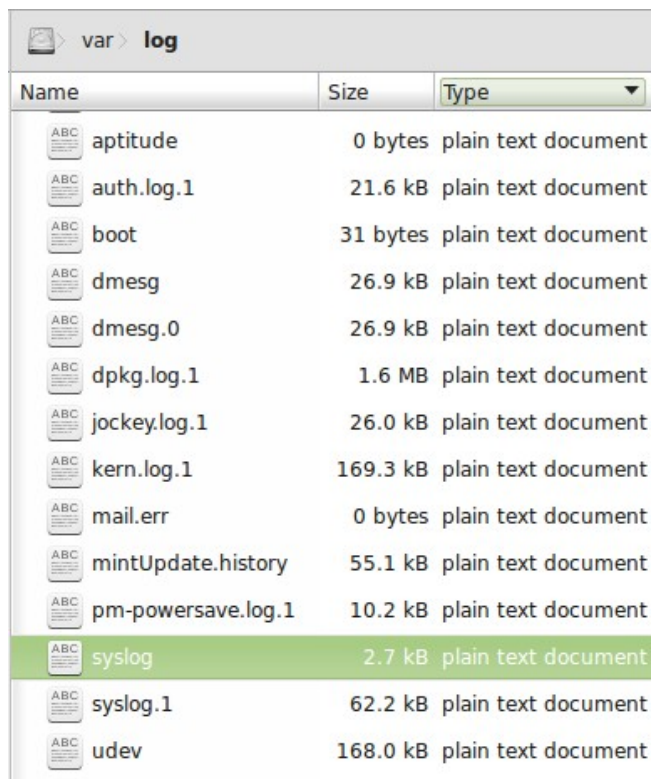
Configurations are in plain text files

The text file
/etc/passwd
contains info
about the users
on your system



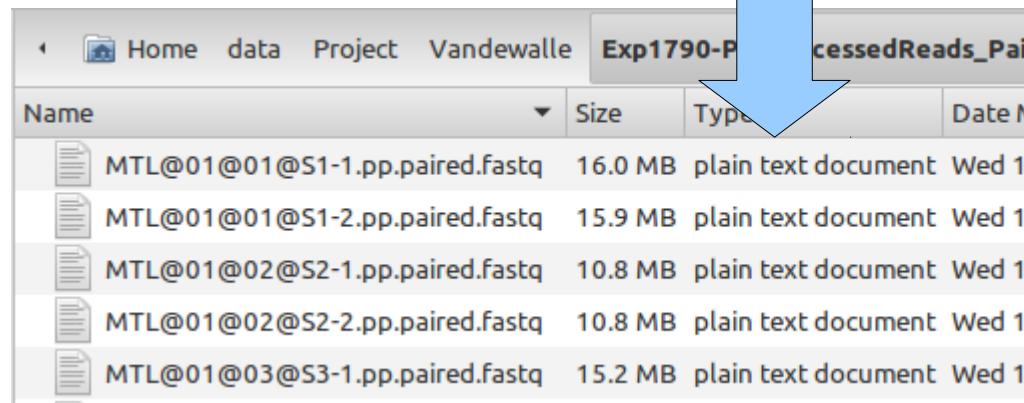
Natural fit of bioinformatics and Linux

Because of this amount of text, Linux comes with a lot of **command line** tools to manipulate / search / analyse text files.



Name	Size	Type
aptitude	0 bytes	plain text document
auth.log.1	21.6 kB	plain text document
boot	31 bytes	plain text document
dmesg	26.9 kB	plain text document
dmesg.0	26.9 kB	plain text document
dpkg.log.1	1.6 MB	plain text document
jockey.log.1	26.0 kB	plain text document
kern.log.1	169.3 kB	plain text document
mail.err	0 bytes	plain text document
mintUpdate.history	55.1 kB	plain text document
pm-powersave.log.1	10.2 kB	plain text document
syslog	2.7 kB	plain text document
syslog.1	62.2 kB	plain text document
udev	168.0 kB	plain text document

Similar to Bioinformatics, which stores data also predominantly in **large text** files.



Name	Size	Type	Date M
MTL@01@01@S1-1.pp.paired.fastq	16.0 MB	plain text document	Wed 1
MTL@01@01@S1-2.pp.paired.fastq	15.9 MB	plain text document	Wed 1
MTL@01@02@S2-1.pp.paired.fastq	10.8 MB	plain text document	Wed 1
MTL@01@02@S2-2.pp.paired.fastq	10.8 MB	plain text document	Wed 1
MTL@01@03@S3-1.pp.paired.fastq	15.2 MB	plain text document	Wed 1

Natural fit of bioinformatics and Linux

All configurations of the system or programs in Linux are stored in **text files**.

The main tool to process these text files is the **terminal**, a program that gives you but a prompt.

Where does the terminal originates from?



Using the terminal as a solution

Before the rise of the nice desktops, users had only this:

Press ctrl + alt + F1

This means – users needed to navigate around on their computers, read files, create files, print, run programs, play games,... **all from the command line.**

And yes, this is possible.



Using the terminal as a solution

Before the rise of the nice desktops, users had only this:

Press ctrl + alt + F1

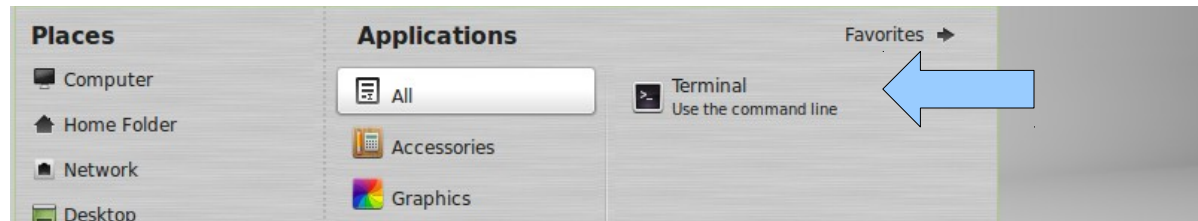


(Lesson one in using Linux. Tux is friendly, but strict.)

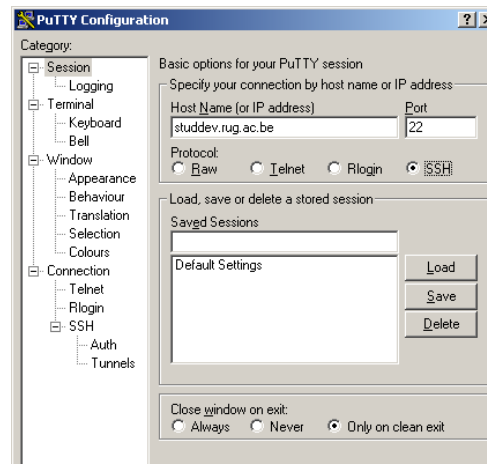
Press ctrl + alt + F7

More practical: use a terminal program

- Open a terminal program using the menu, and execute commands on your computer:

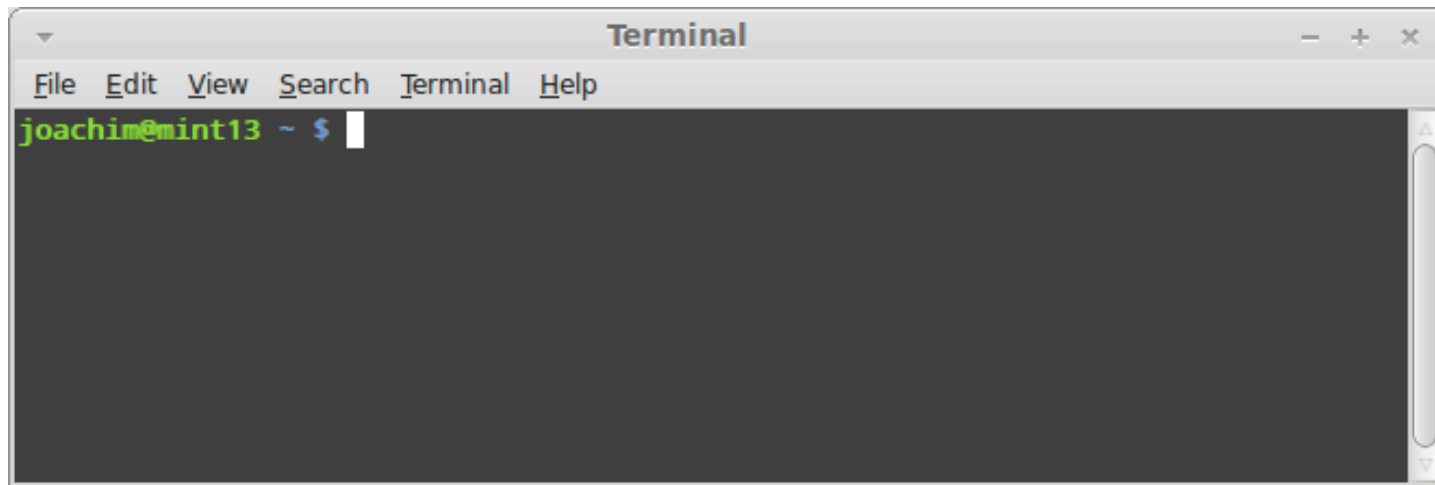


- Also possible: connect to a terminal over the internet, e.g. using Putty program installed on a Windows machine.



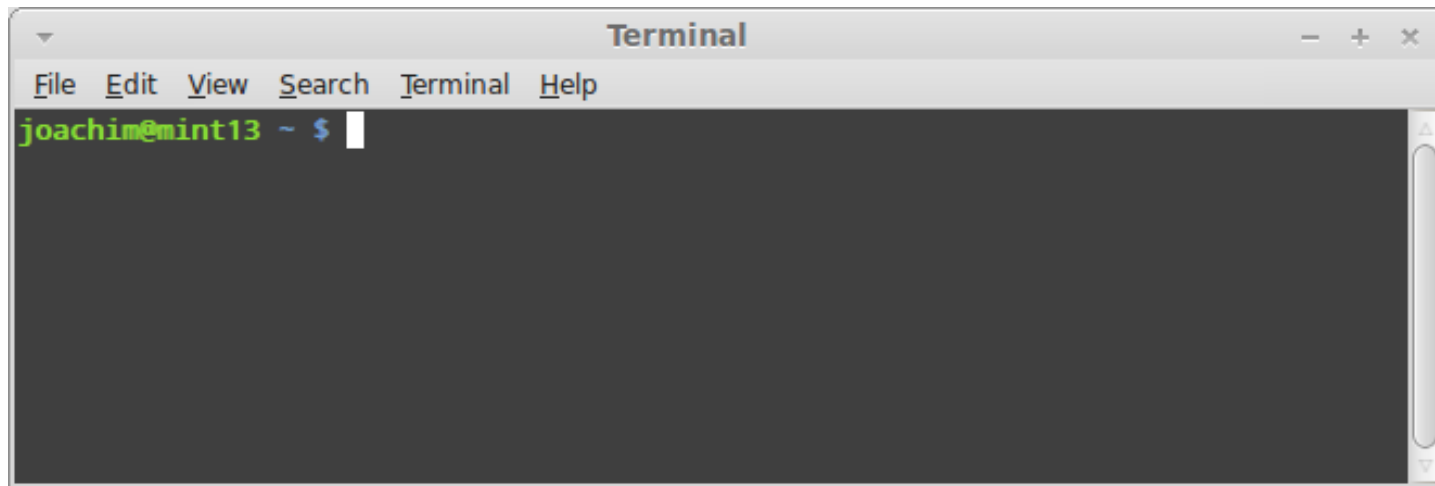
Your first steps in a terminal

Starting a terminal, we end up with this:



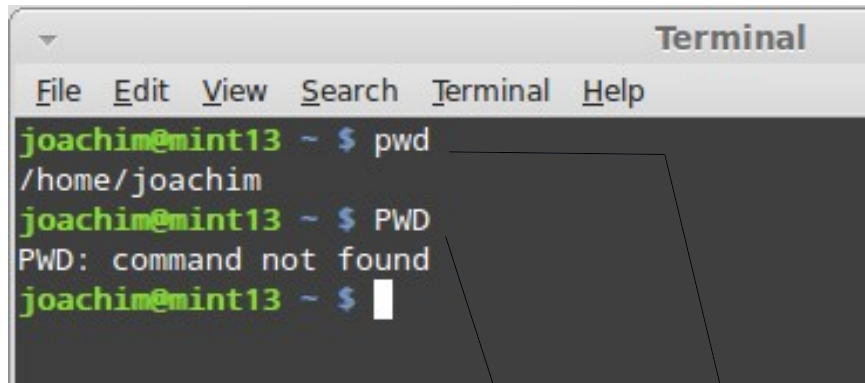
A terminal accepts only input from the keyboard, the command line. You can only type. What you type gets interpreted by the program **Bash**, which will then do what you ask.

Few important things



- A command line is always positioned **somewhere in the file system**.
- What you type is **case-sensitive**
- The prompt line can be customized, but by default it shows:
 - Username
 - @

Check where your shell is positioned



```
Terminal
File Edit View Search Terminal Help
joachim@mint13 ~ $ pwd
/home/joachim
joachim@mint13 ~ $ PWD
PWD: command not found
joachim@mint13 ~ $
```

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is "joachim@mint13 ~ \$". The user enters "pwd", and the output is "/home/joachim". The user then enters "PWD", and the output is "PWD: command not found". The prompt returns to "joachim@mint13 ~ \$".

...

After typing your command,
press **<enter>** to execute
a command.

pwd = print working directory

Navigating in the file system

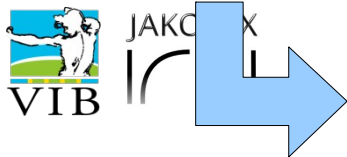
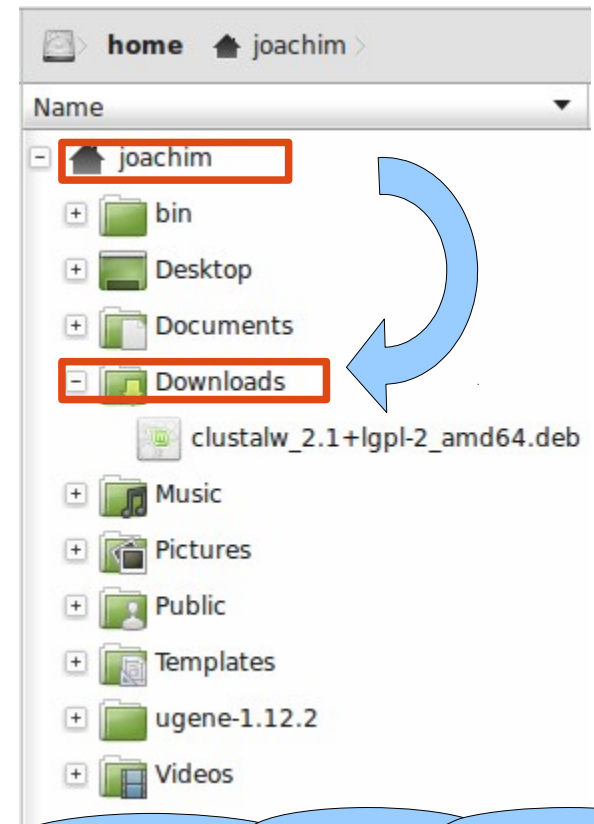
```
Terminal
File Edit View Search Terminal Help
joachim@mint13 ~ $ ls
bin      Documents Music      Public  ugene-1.12.2
Desktop  Downloads Pictures  Templates Videos
joachim@mint13 ~ $ cd Downloads
joachim@mint13 ~/Downloads $
```

ls = list contents of current directory

cd = change to this directory

The word after 'cd' matches a directory.
We tell 'cd' to go to this directory. This
Additional word is called an **argument**.

The result is that the prompt
has changed position from:

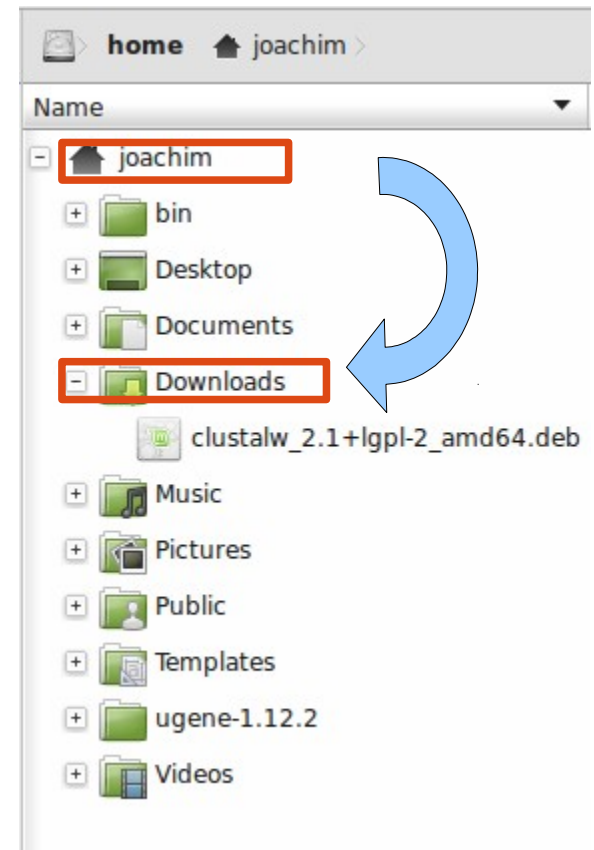


What will happen if we type 'ls' now?

Navigating in the file system

The result is that the prompt has changed position from:

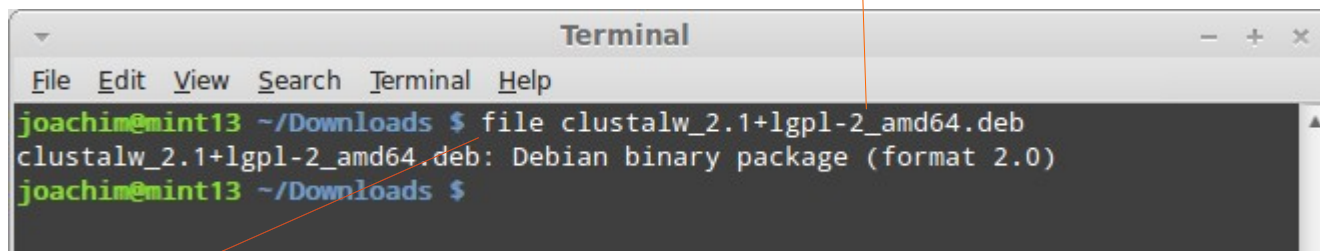
```
Terminal
File Edit View Search Terminal Help
joachim@mint13 ~ $ ls
bin      Documents Music      Public  ugene-1.12.2
Desktop  Downloads Pictures  Templates Videos
joachim@mint13 ~ $ cd Downloads
joachim@mint13 ~/Downloads $ ls
clustalw_2.1+lgpl-2_amd64.deb
joachim@mint13 ~/Downloads $
```



Running a command on a file

We can provide a file name as an **argument** to a command. In the Downloads directory, we can for example check a file type with the command **file**.

Argument points to a file



```
Terminal
File Edit View Search Terminal Help
joachim@mint13 ~/Downloads $ file clustalw_2.1+lgpl-2_amd64.deb
clustalw_2.1+lgpl-2_amd64.deb: Debian binary package (format 2.0)
joachim@mint13 ~/Downloads $
```

file = is a command, it shows the file type of the file passed as argument

You never walk alone

How to know the commands, and which arguments they need?

A Linux system comes with batteries included: only they are called **man-pages** (manual).

The program **man** displays the manual for the program provided as argument.

For example: the manual of **ls**.

You never walk alone

```
Terminal
File Edit View Search Terminal Help
joachim@mint13 ~/Downloads $ man ls
```

Execute the program 'man'
with argument 'ls', meaning:
show me the manual of ls

```
Terminal
File Edit View Search Terminal Help
LS(1) User Commands LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILES (the current directory by default).
    Sort entries alphabetically if none of -cftuvSUX nor --sort is speci-
    fied.

    Mandatory arguments to long options are mandatory for short options
    too.

    -a, --all
        do not ignore entries starting with .

    -A, --almost-all
        do not list implied . and ..

    --author
        Manual page ls(1) line 1 (press h for help or q to quit)
```

E.g. Bowtie has also a manual

```
Terminal
File Edit View Search Terminal Help
BOWTIE(1) User Commands
BOWTIE(1)

NAME
    bowtie - ultrafast memory-efficient short read aligner

DESCRIPTION
    Usage:
        bowtie [options]* <ebwt> {-1 <m1> -2 <m2> | --12 <r> | <s>} [<hit>]

        <m1>    Comma-separated list of files containing upstream mates (or the sequences themselves, if -c is set) paired with mates in <m2>

        <m2>    Comma-separated list of files containing downstream mates (or the sequences themselves if -c is set) paired with mates in <m1>

        <r>      Comma-separated list of files containing Crossbow-style reads. Can be a mixture of paired and unpaired. Specify "-" for stdin.

        <s>      Comma-separated list of files containing unpaired reads, or the sequences themselves, if -c is set. Specify "-" for stdin.

        <hit>    File to write hits to (default: stdout)

    Input:
        -q      query input files are FASTQ .fq/.fastq (default)
        -f      query input files are (multi-)FASTA .fa/.mfa
        -r      query input files are raw one-sequence-per-line
        -c      query sequences given on cmd line (as <mates>, <singles>)
        -C      reads and index are in colorspace
        -Q/--quals <file>

Manual page bowtie(1) line 1 (press h for help or q to quit)
```

Usage tips for the program man

↑ and ↓	scroll up and down
PgUp	previous page
PgDown or space	next page
< and >	begin and end of the text file
/	search (forward)
n	next search hit
q	to exit

(The command man uses less under the hood to display the manual page.)

Using the terminal: copy/pasting

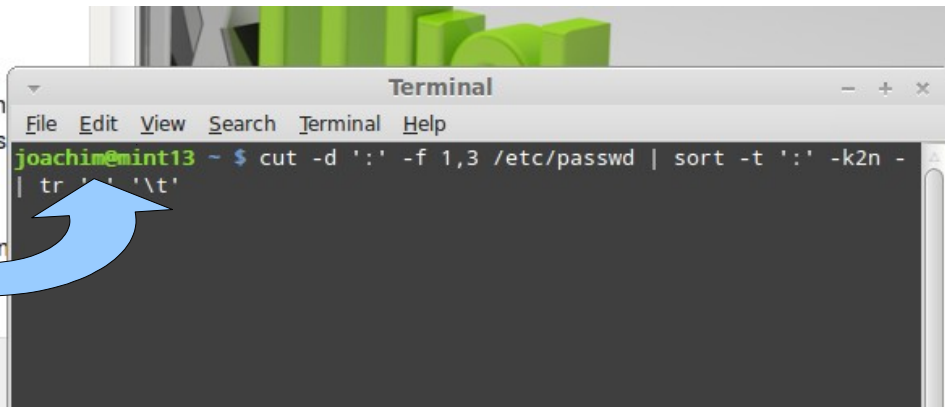
- Normal copy/paste: hold shift + ctrl + c / v
- Select text somewhere (e.g. in browser), and **click with the scroll wheel** in your terminal.

1. Display Username and UID sorted by UID Using cut, sort and tr

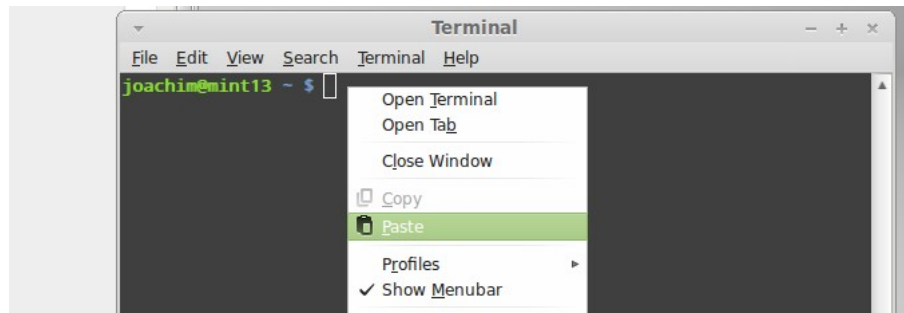
cut command is used to extract specific part of a file. The following example cuts the username and UID from /etc/passwd file, and sort the output using sort command using username as key and ":" as a delimiter.

As a part of formatting the output, you can use any other character to display username and UID. Using tr command you can convert ":" to any other character.

```
$ cut -d ':' -f 1,3 /etc/passwd | sort -t ':' -k2n - | tr ':' '\t'
root      0
```



- Copy and paste by right-clicking and selecting from the menu



Fine-tuning commands behaviour

- Reading man pages you can find how to use the **arguments and options**.
- Setting these influences the way commands behave. **Options** precede arguments, and are separated from the command (and from each other) by **spaces**. They usually start with '-' or '--'.
- For example:

```
$ ls -l /bin
```

From now on this will be the notation for command line instructions as a normal user

Program (first thing you type should always be a program)

Option: you want 'ls' to change default behaviour: -l adds more info ('long').

Argument: on what content (file, directory,...) should ls operate.

Help! The most used option

Another way to get help, besides the man pages, is to invoke the option '**--help**'. Nearly every command has this option.

```
joachim@joachim-VirtualBox ~ $ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                do not ignore entries starting with .
  -A, --almost-all        do not list implied . and ..
      --author              with -l, print the author of each file
  -b, --escape              print C-style escapes for nongraphic characters
      --block-size=SIZE    scale sizes by SIZE before printing them. E.g.,
                           '--block-size=M' prints sizes in units of
                           1,048,576 bytes. See SIZE format below.
  -B, --ignore-backups     do not list implied entries ending with ~
  -c                        with -lt: sort by, and show, ctime (time of last
                           modification of file status information)
                           with -l: show ctime and sort by name
                           otherwise: sort by ctime, newest first
```

Short and long options

Some options have two names: a short and a long name. **Short** are one character long, and start with a '-'. **Long** options are usually a word, and are preceded by '--'.

```
joachim@joachim-VirtualBox ~ $ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
-a, --all                do not ignore entries starting with .
-A, --almost-all        do not list implied . and ..
    --author              with -l, print the author of each file
-b, --escape              print C-style escapes for nongraphic characters
    --block-size=SIZE    scale sizes by SIZE before printing them.  E.g.,
                        '--block-size=M' prints sizes in units of
                        1,048,576 bytes.  See SIZE format below.
-B, --ignore-backups     do not list implied entries ending with ~
-c                        with -lt: sort by, and show, ctime (time of last
                        modification of file status information)
                        with -l: show ctime and sort by name
                        otherwise: sort by ctime, newest first
```

Short options peculiarities

- You can add multiple options to the command.
(the given order is seldom important)

```
$ ls -l -t /bin
```

```
$ ls -t -l /bin
```

- Using short options allow you to **combine** several options in one string.

```
$ ls -r -l -t
```

```
$ ls -rtl
```

Long options

- 'Long' options consist of -- (two dashes) followed by the name of the option (string):
\$ ls --recursive
- Long options **cannot be combined** like their 'short' counterparts

Options can also have arguments

- For example, show the contents of the directory, sorted by size of the files.
- We use the option `--sort` for this: the argument to sort must follow the option:

```
$ ls --sort=size /bin
```

- the '=' sign is optional

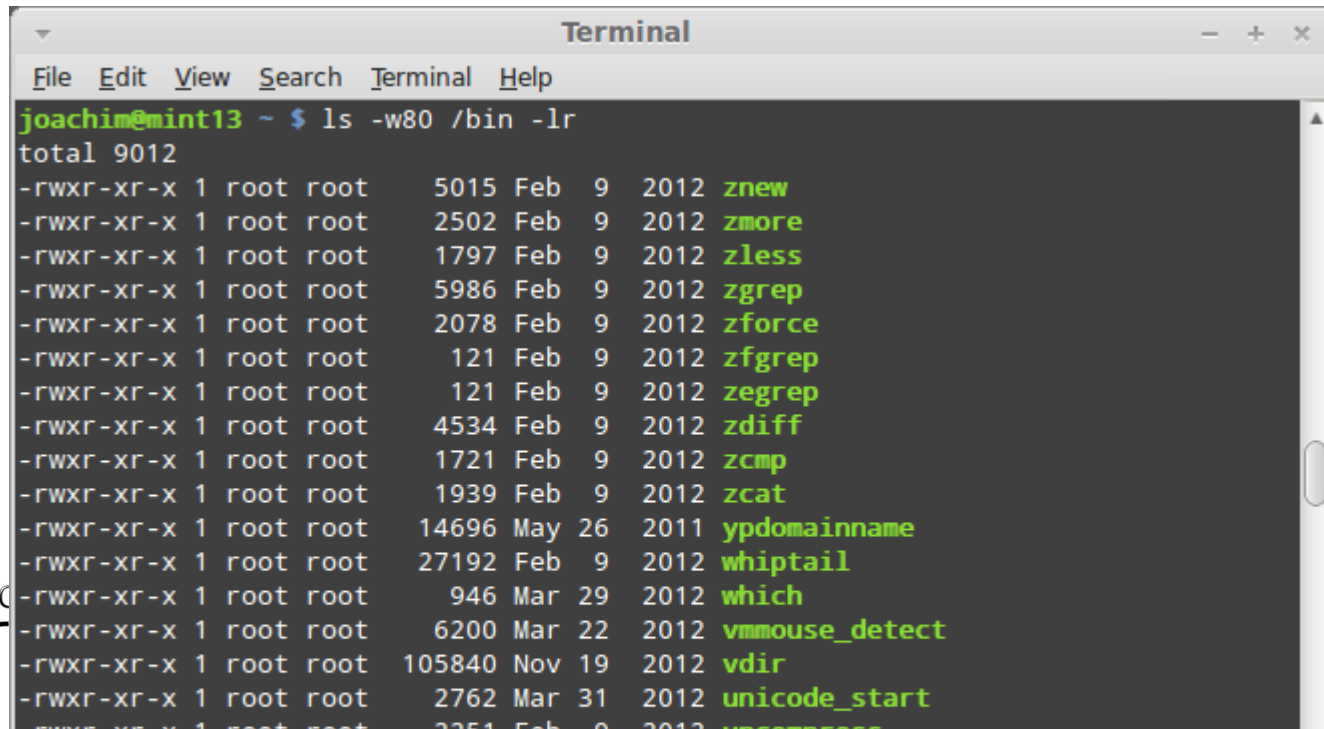
- An example of an argument to a short option:

```
$ ls -w 80 /bin
```

- the space between option name and argument is optional

Different combinations of options

```
$ ls -lir -w 80 /bin
$ ls -rlw 80 /bin
$ ls -wrl 80 /bin      # NOK
$ ls -w80 /bin -lir
```



A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help) and window controls. The prompt is "joachim@mint13 ~". The command entered is "ls -w80 /bin -lir". The output shows a list of files in /bin with permissions, owner, group, size, date, and filename. The filenames are highlighted in green.

```
joachim@mint13 ~ $ ls -w80 /bin -lir
total 9012
-rwxr-xr-x 1 root root    5015 Feb  9  2012 znew
-rwxr-xr-x 1 root root    2502 Feb  9  2012 zmore
-rwxr-xr-x 1 root root    1797 Feb  9  2012 zless
-rwxr-xr-x 1 root root    5986 Feb  9  2012 zgrep
-rwxr-xr-x 1 root root    2078 Feb  9  2012 zforce
-rwxr-xr-x 1 root root     121 Feb  9  2012 zfgrep
-rwxr-xr-x 1 root root     121 Feb  9  2012 zegrep
-rwxr-xr-x 1 root root    4534 Feb  9  2012 zdiff
-rwxr-xr-x 1 root root    1721 Feb  9  2012 zcmp
-rwxr-xr-x 1 root root    1939 Feb  9  2012 zcat
-rwxr-xr-x 1 root root   14696 May 26  2011 ypdomainname
-rwxr-xr-x 1 root root   27192 Feb  9  2012 whiptail
-rwxr-xr-x 1 root root     946 Mar 29  2012 which
-rwxr-xr-x 1 root root    6200 Mar 22  2012 vmmouse_detect
-rwxr-xr-x 1 root root  105840 Nov 19  2012 vdir
-rwxr-xr-x 1 root root    2762 Mar 31  2012 unicode_start
-rwxr-xr-x 1 root root     2351 Feb  9  2012 vncserver
```

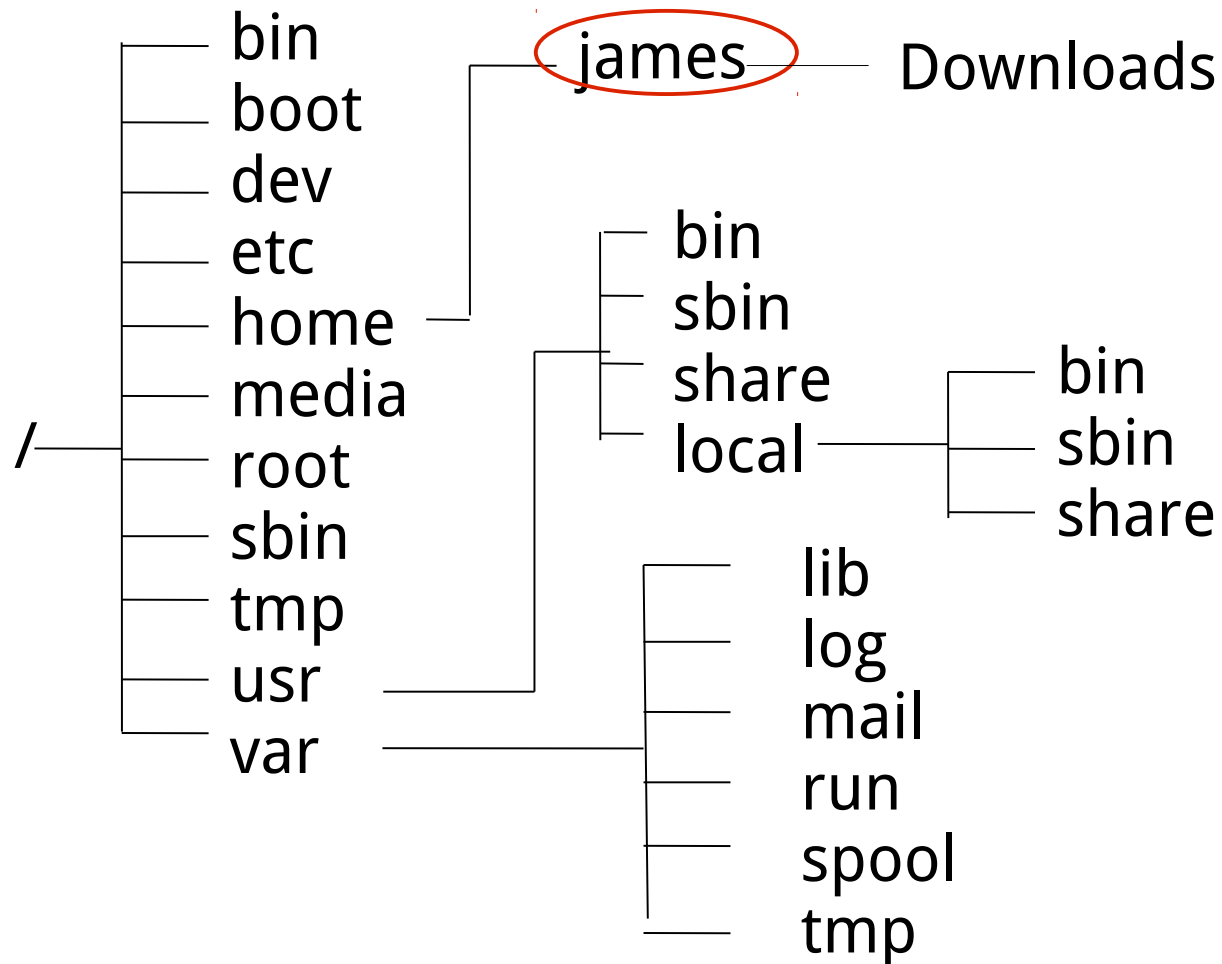
Some basic commands

Command	Explanation
pwd	Print working directory
ls	Print content of directory
cd	Change directory
cat	Print the contents of a file
cp	Copy a file
mv	Move a file
rm	Remove a file
less	Read the contents of a file
clear	Clear the terminal screen
head	Show the first 10 lines of a file
tail	Show the last 10 lines of a file
nano	Text editor, to modify text files
wget	Download a file from an URL

Exercise: gentle intro to the command line

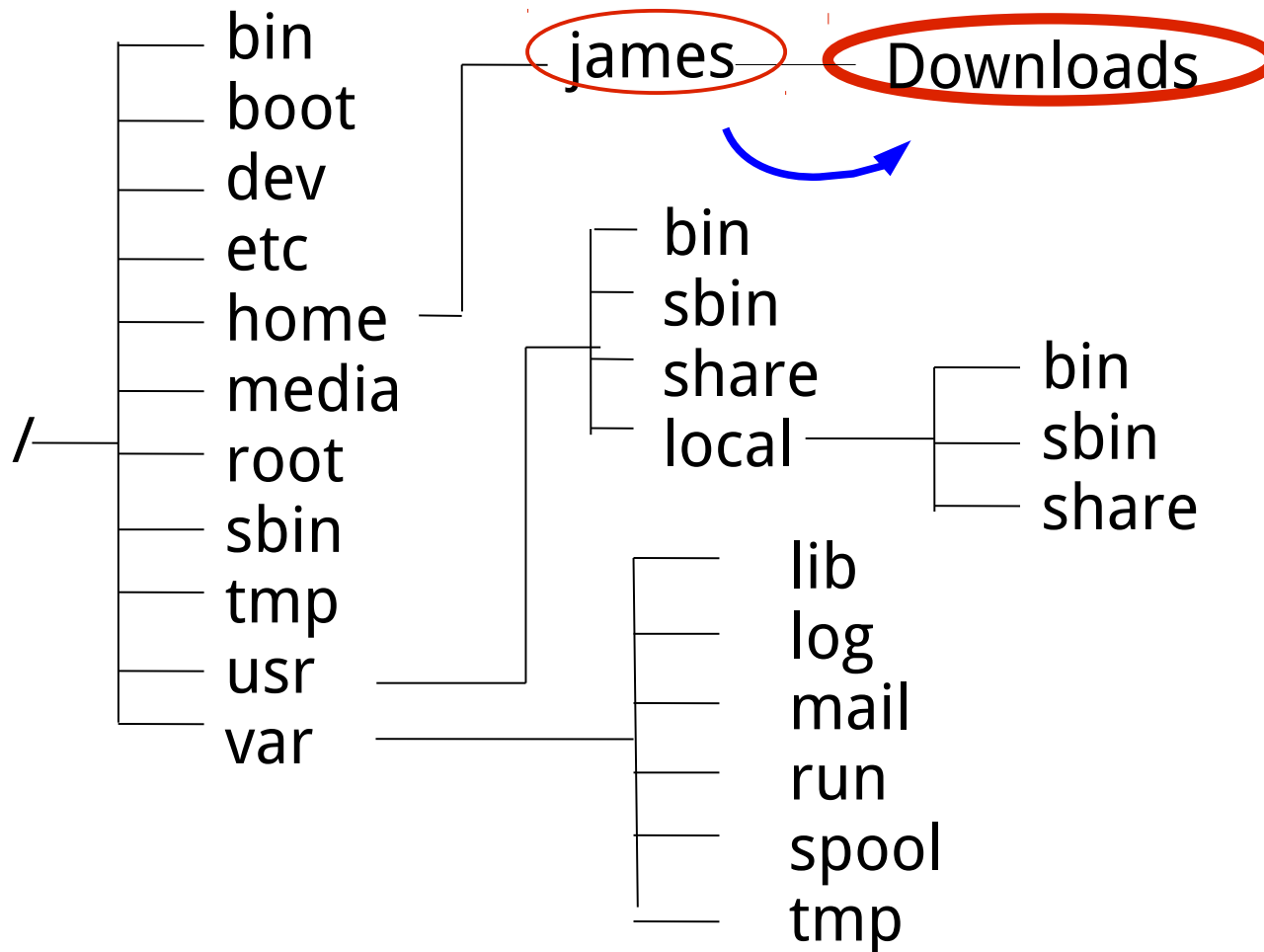
→ [Exercise link](#)

Paths and the working directory



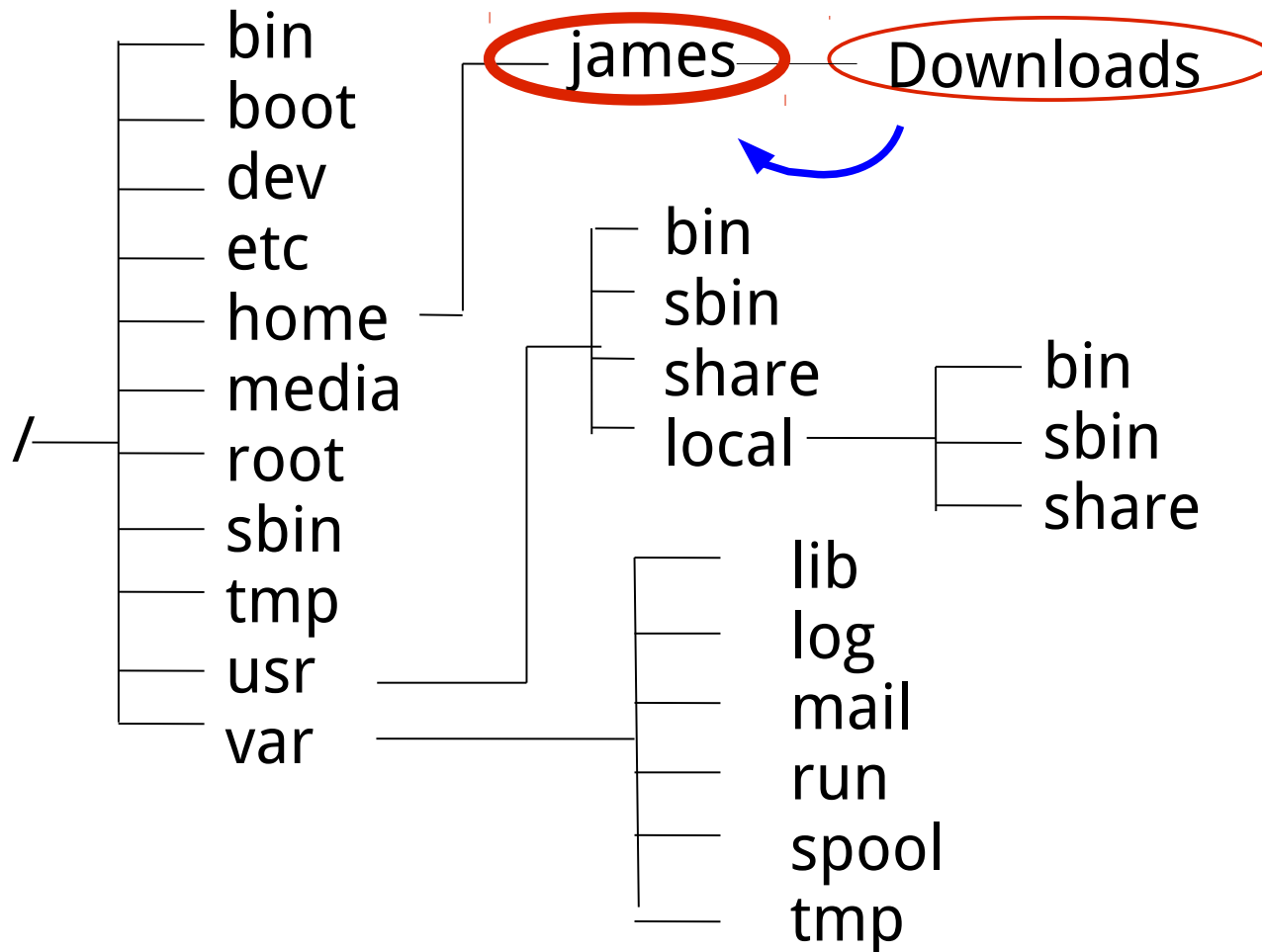
Paths and the working directory

```
~ $ cd Downloads
```



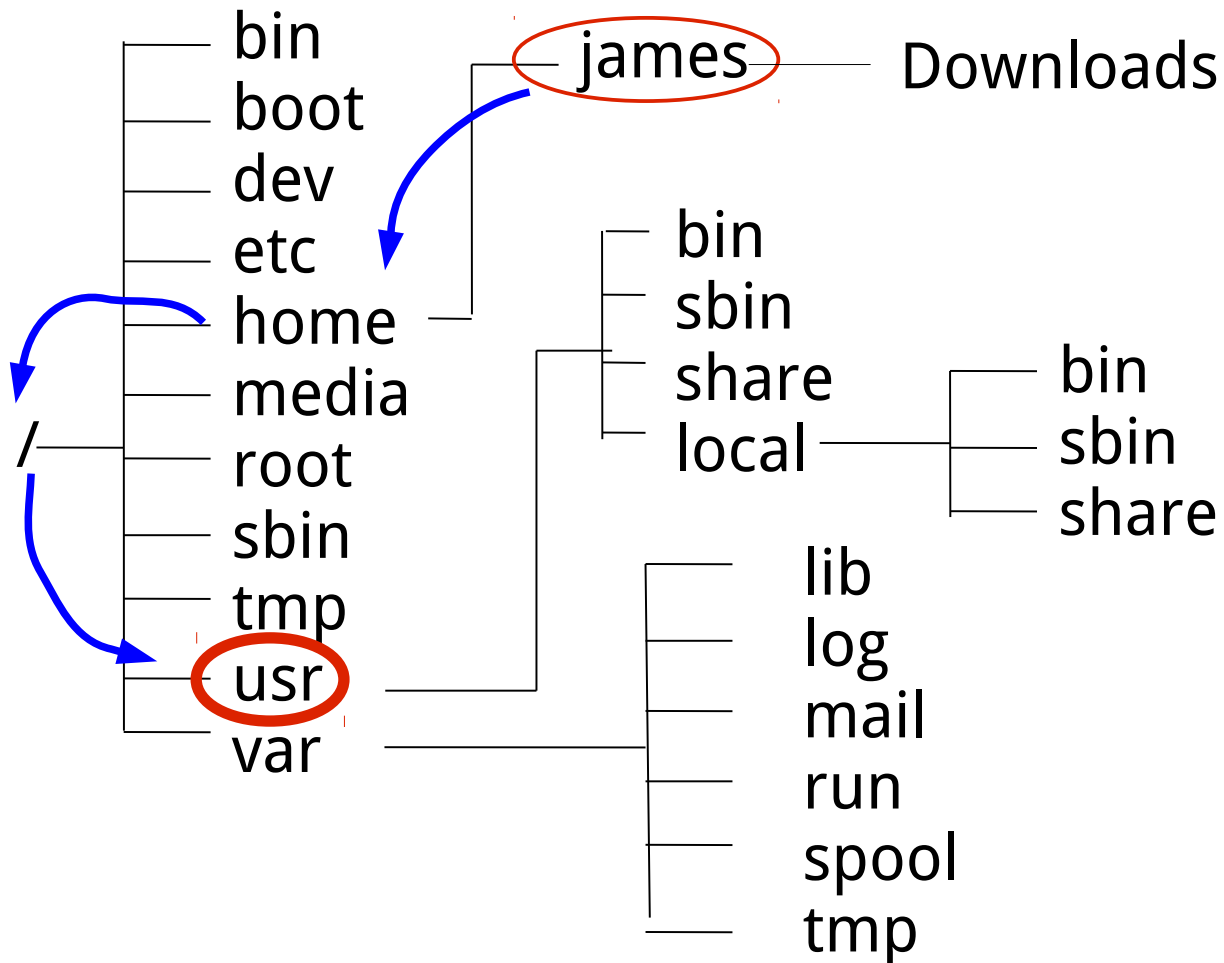
Paths and the working directory

```
~/Downloads $ cd ..
```

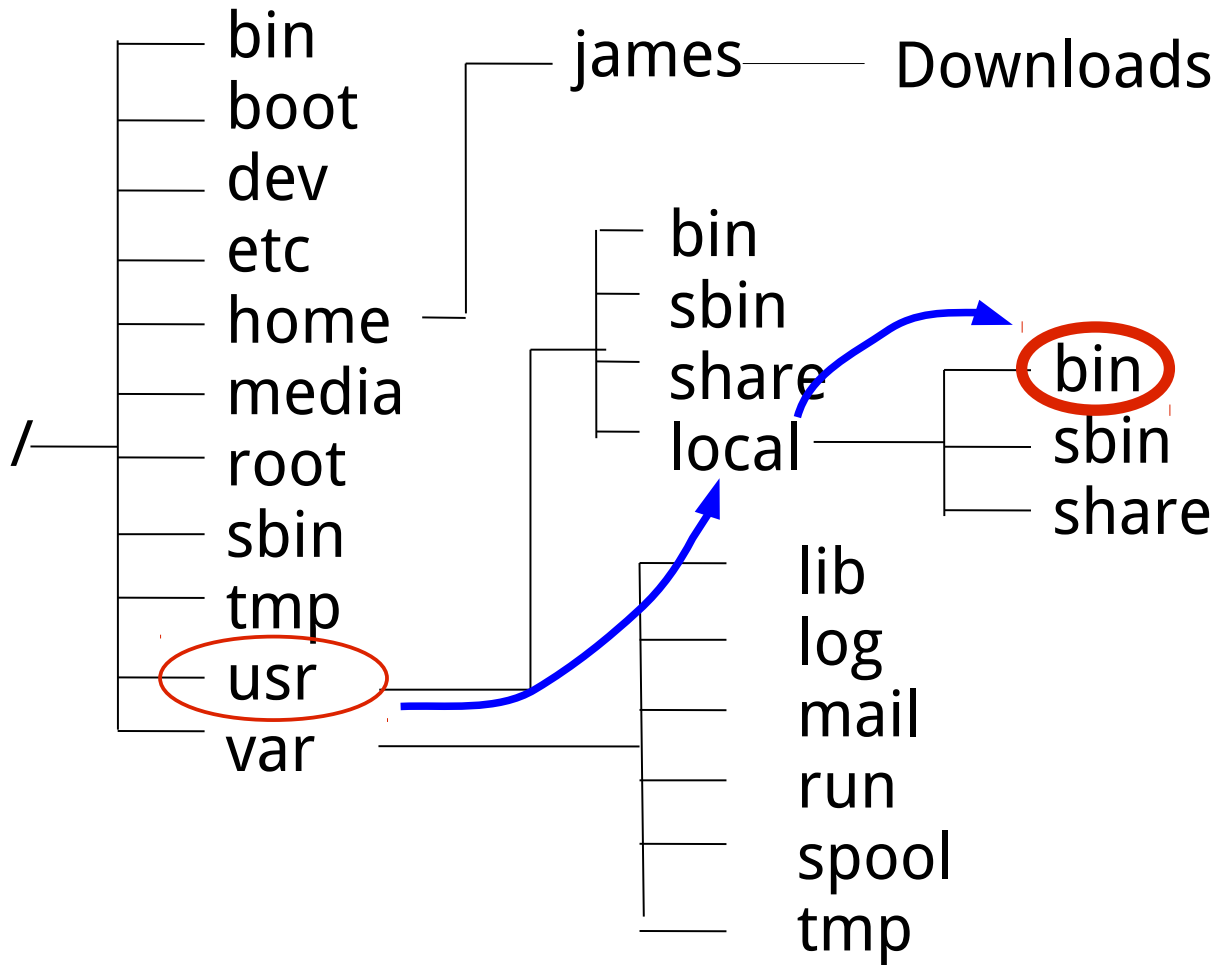


Paths and the working directory

```
~ $ cd ../../usr
```

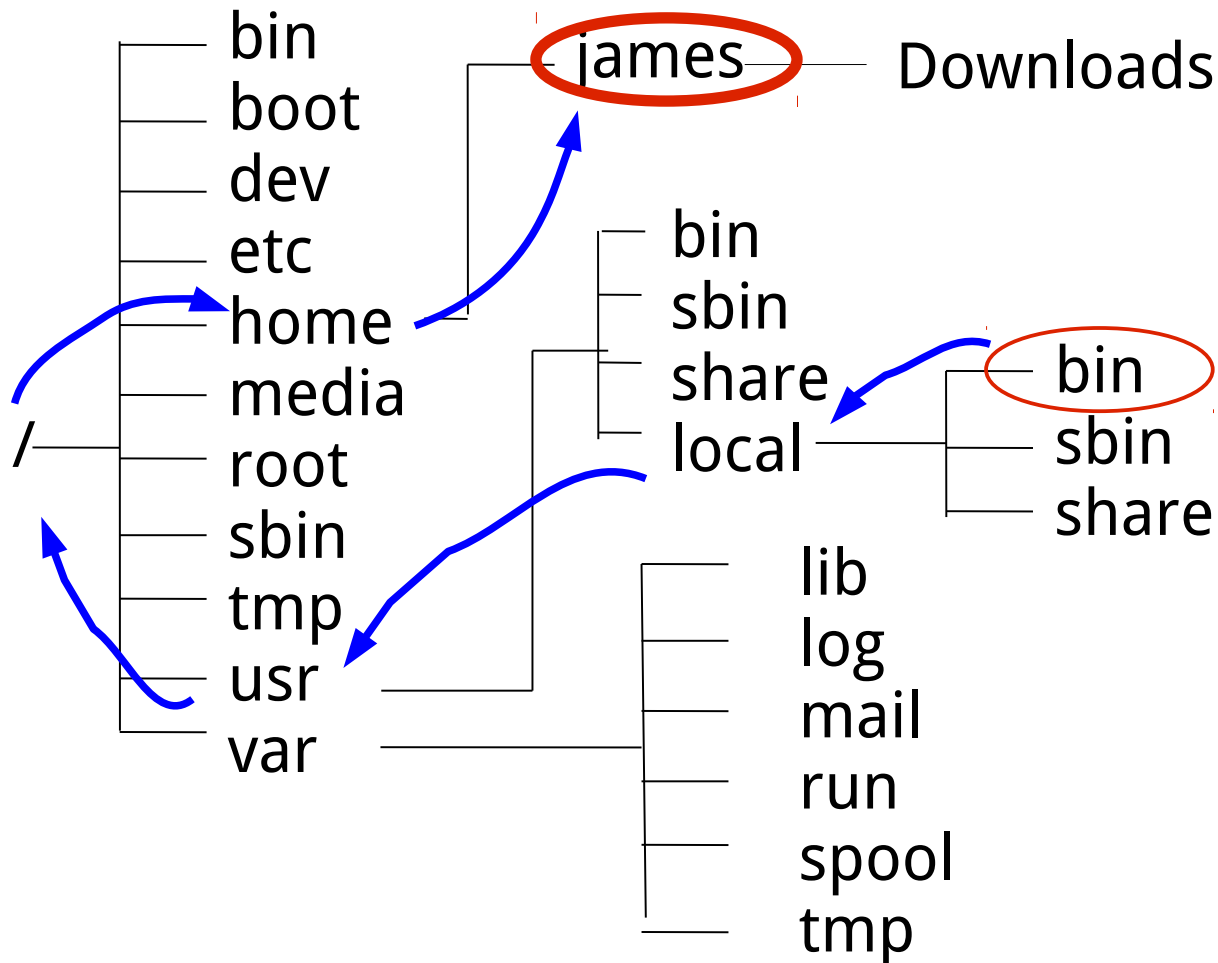


```
/usr $ cd local/bin
```



Paths and the working directory

```
/usr/local/bin $ cd ../../../../home/james
```



Relative paths

The **path** is set up **relative to the current working directory**.

In other words: which steps do we take from the current directory to reach the destination.

```
/usr/local/bin $ cd ../../../../home/james
```

```
/usr $ cd local/bin
```

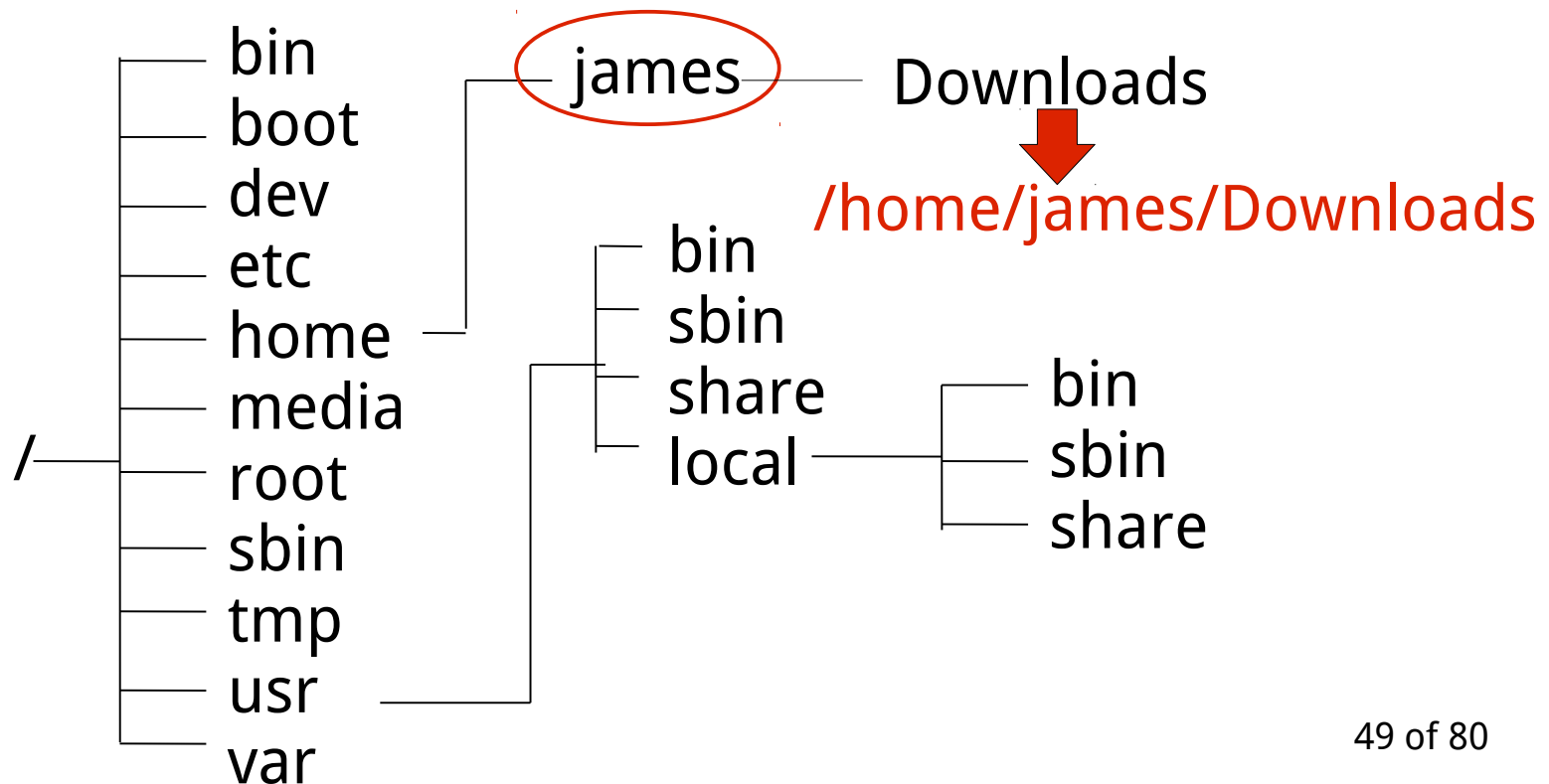
```
~ $ cd ../../usr
```

```
~ $ cd Downloads
```

```
~/Downloads $ cd ..
```

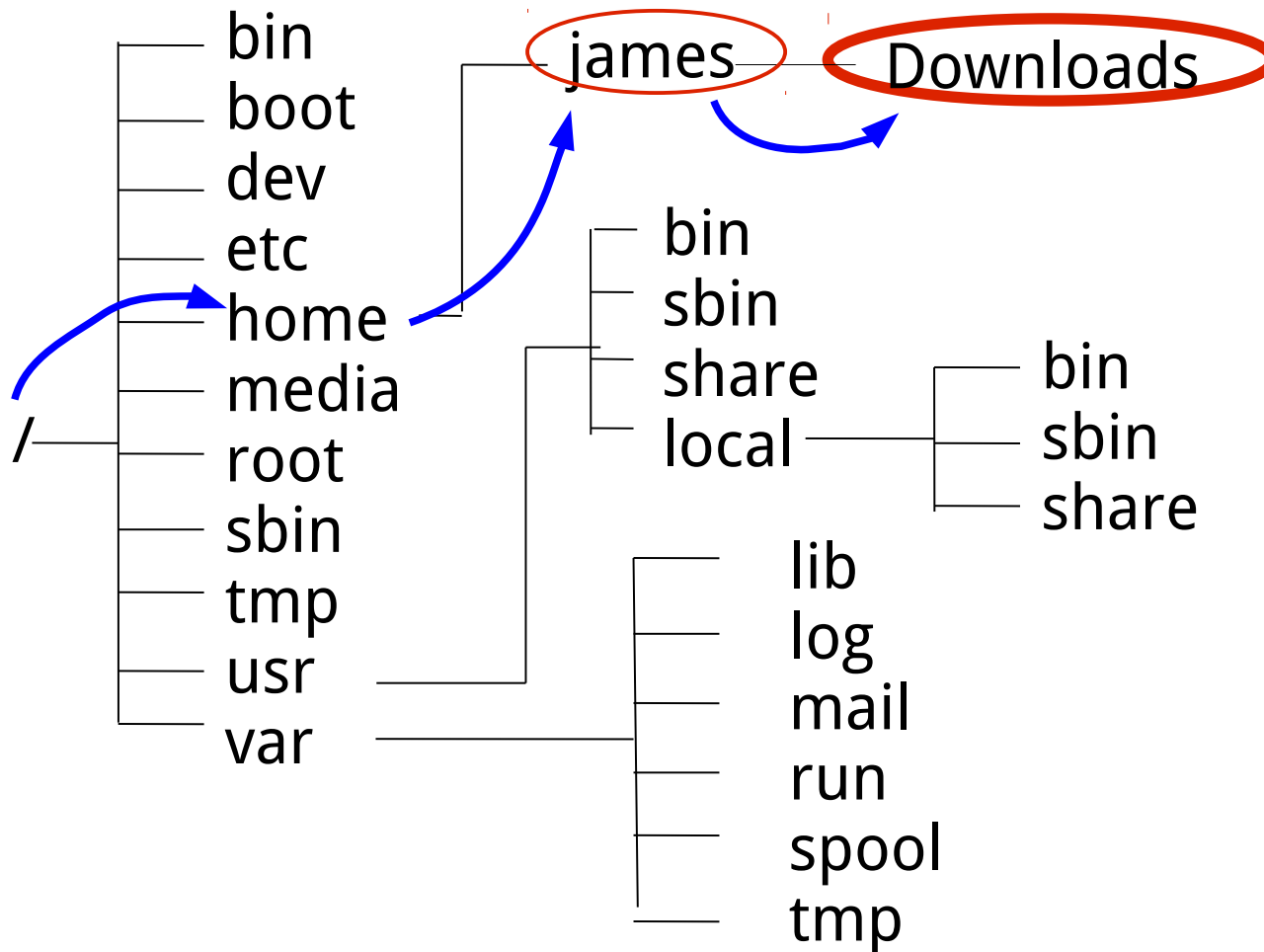

Absolute paths

Sometimes it is more convenient to point to the complete or **absolute** path of the directory, starting from the **root directory**.



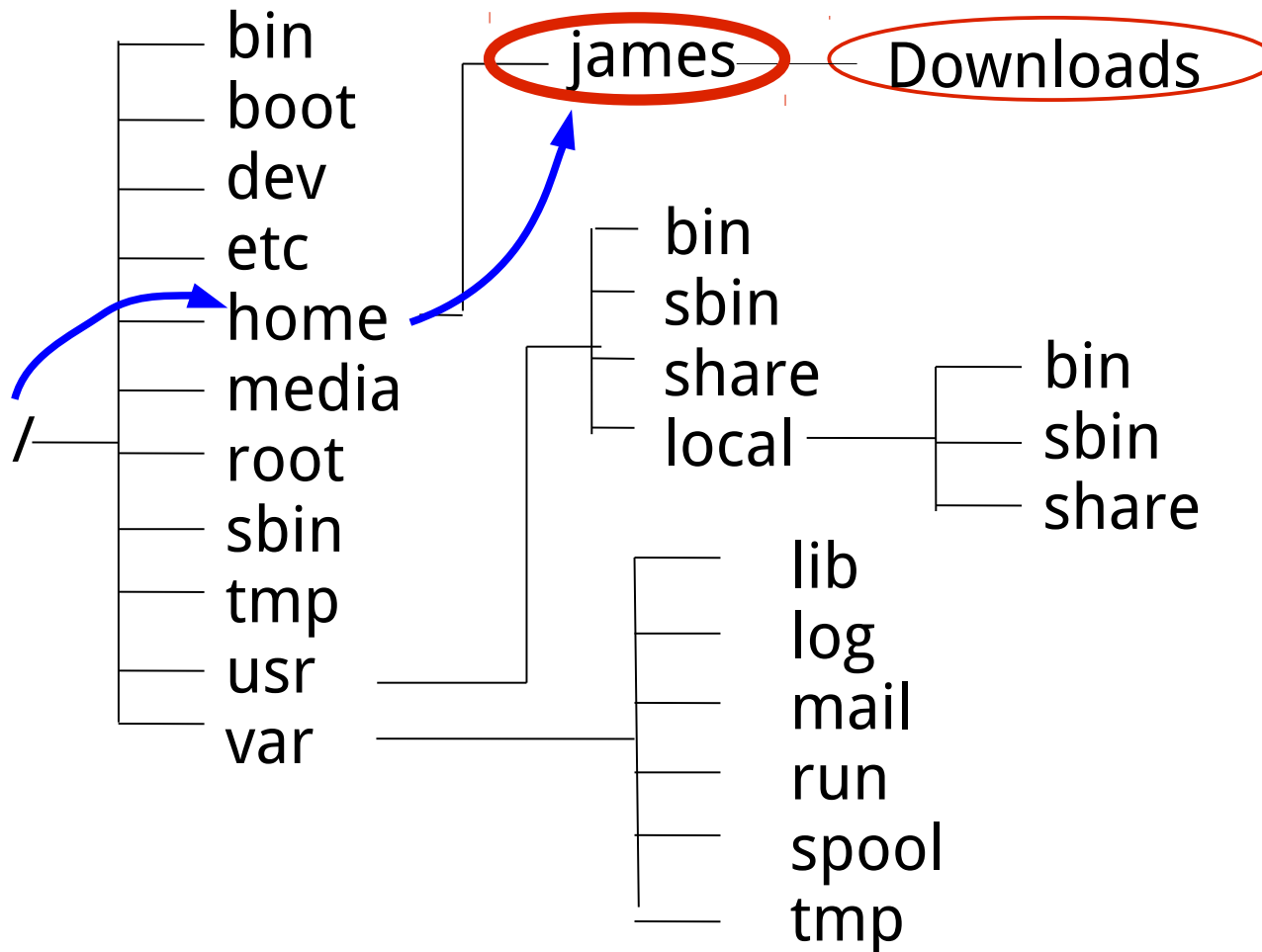
Paths and the working directory

```
~ $ cd /home/james/Downloads
```



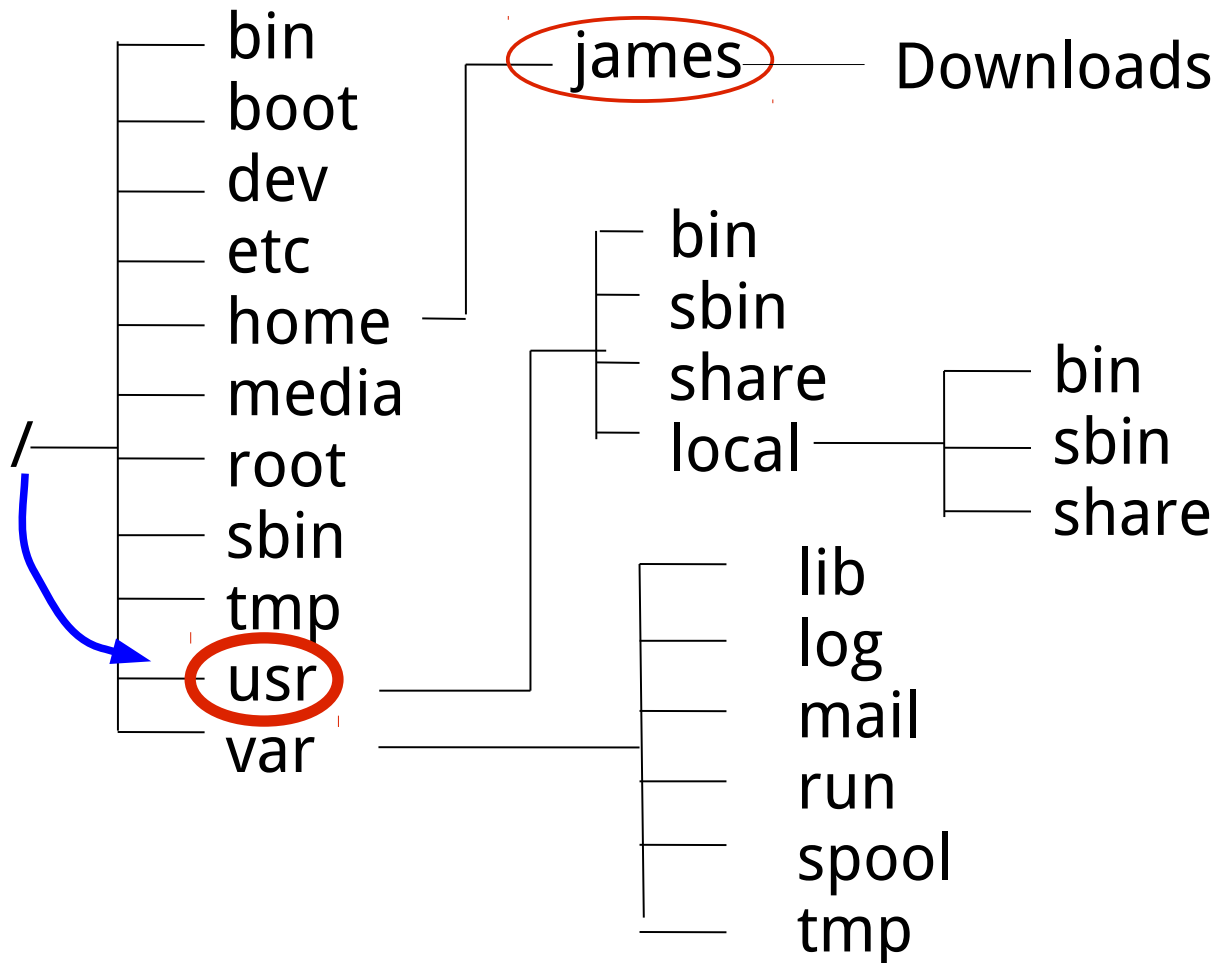
Paths and the working directory

```
~/Downloads $ cd /home/james
```



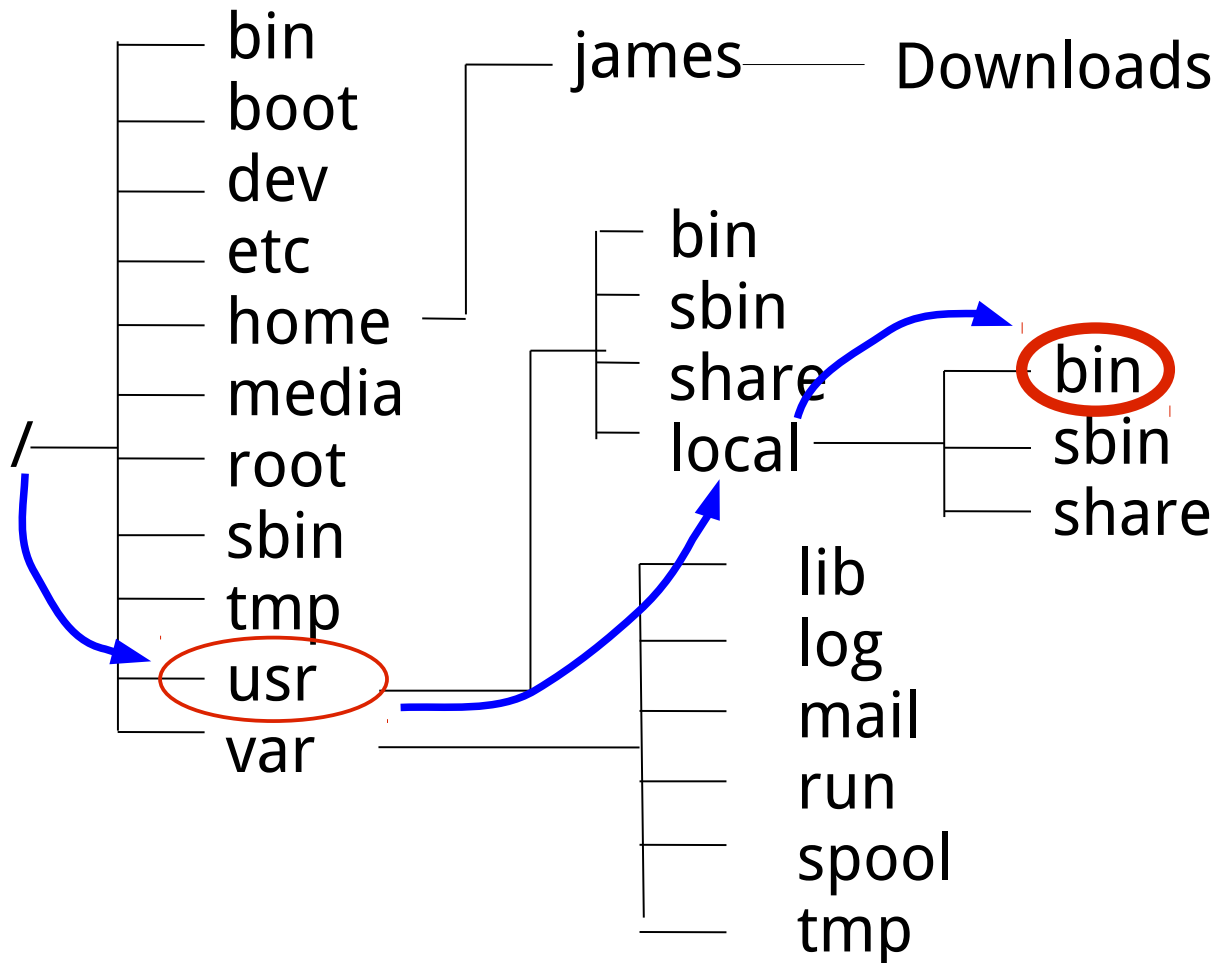
Paths and the working directory

```
~ $ cd /usr
```



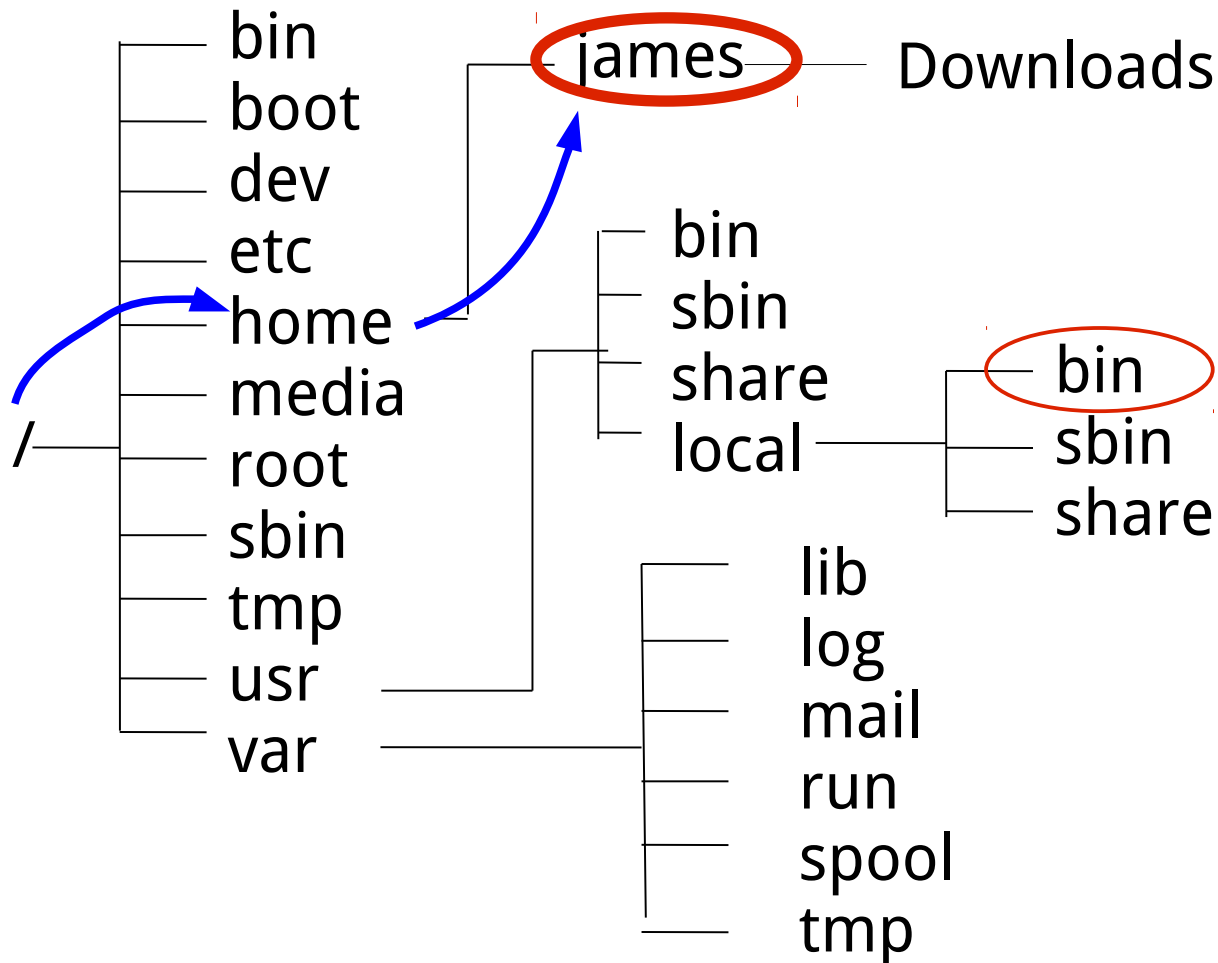
Paths and the working directory

```
/usr $ cd /usr/local/bin
```



Paths and the working directory

```
/usr/local/bin $ cd /home/james
```



Relative versus absolute paths

Relative ↔ absolute (=starts always with **/**, the root)

```
/usr/local/bin $ cd ../../../../home/james
```

```
/usr/local/bin $ cd /home/james
```

```
/usr $ cd local/bin
```

```
/usr $ cd /usr/local/bin
```

```
~ $ cd ../../usr
```

```
~ $ cd /usr
```

```
~ $ cd Downloads
```

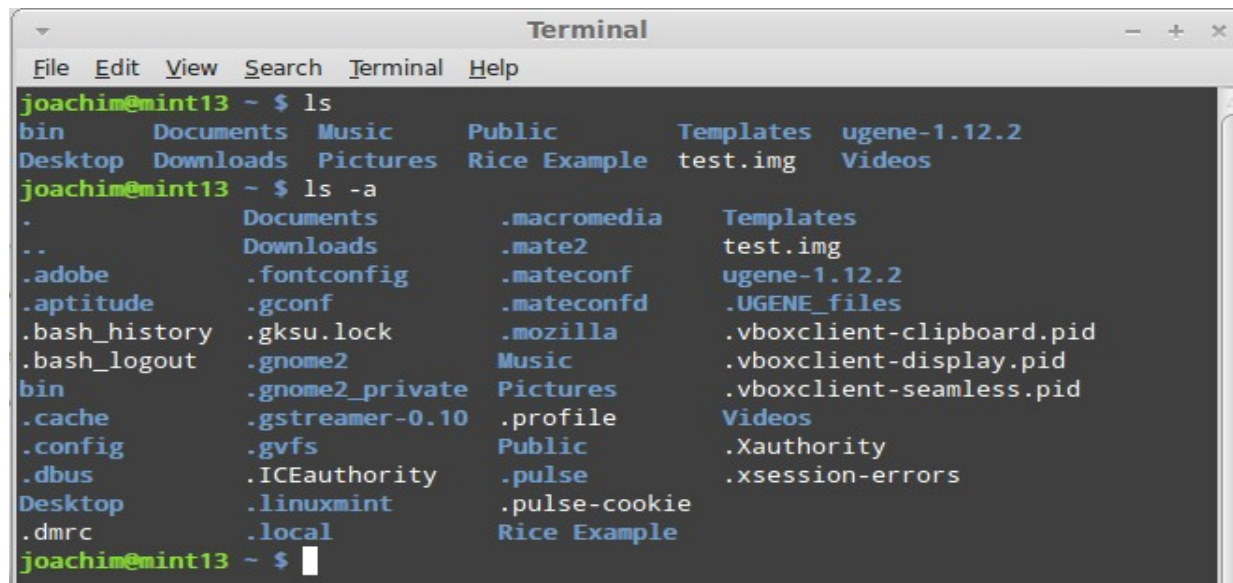
```
~ $ cd /home/james/Downloads
```

```
~/Downloads $ cd ..
```

```
~/Downloads $ cd /home/james
```

Hidden directories

- Compare the output of `ls` versus `ls -a`



A terminal window titled "Terminal" showing the output of two commands. The first command is `ls`, which lists visible directories and files. The second command is `ls -a`, which lists all files and directories, including hidden ones starting with a dot. The hidden files and directories are listed in columns between the visible ones.

```
joachim@mint13 ~ $ ls
bin      Documents Music      Public      Templates  ugene-1.12.2
Desktop  Downloads Pictures    Rice Example test.img    Videos
joachim@mint13 ~ $ ls -a
.         Documents      .macromedia  Templates
..        Downloads      .mate2        test.img
.adobe    .fontconfig    .mateconf     ugene-1.12.2
.apptitude .gconf         .mateconfd    .UGENE_files
.bash_history .gksu.lock     .mozilla      .vboxclient-clipboard.pid
.bash_logout .gnome2        Music         .vboxclient-display.pid
bin        .gnome2_private Pictures       .vboxclient-seamless.pid
.cache     .gstreamer-0.10 .profile      Videos
.config    .gvfs          Public        .Xauthority
.dbus      .ICEauthority  .pulse        .xsession-errors
Desktop    .linuxmint     .pulse-cookie
.dmrc      .local         Rice Example
joachim@mint13 ~ $
```

- Check with the file manager your home.
- **Hidden** files and folders start with '.'. They are not shown by default.

Moving data around

- Copy files:

```
$ cp <what> <to where>
```

- Move files:

```
$ mv <what> <to where>
```

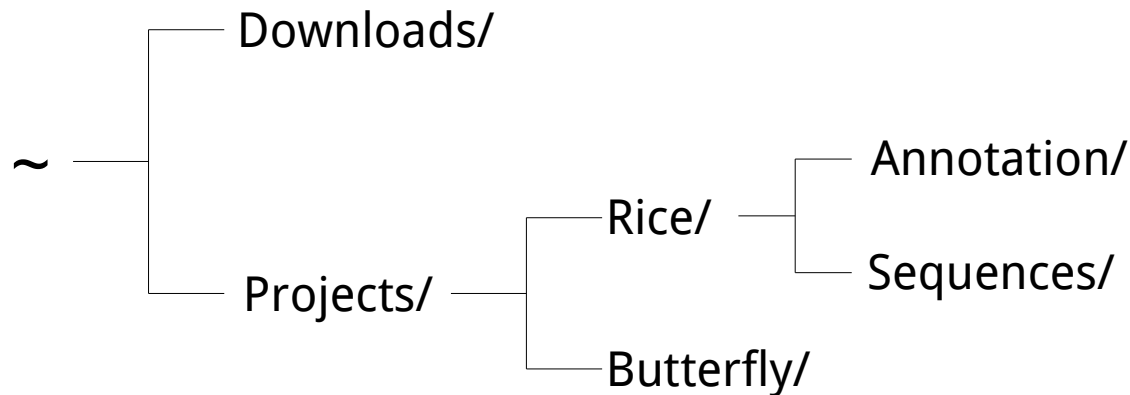
- Remove files:

```
$ rm <filename>
```

Absolute or relative paths to the files

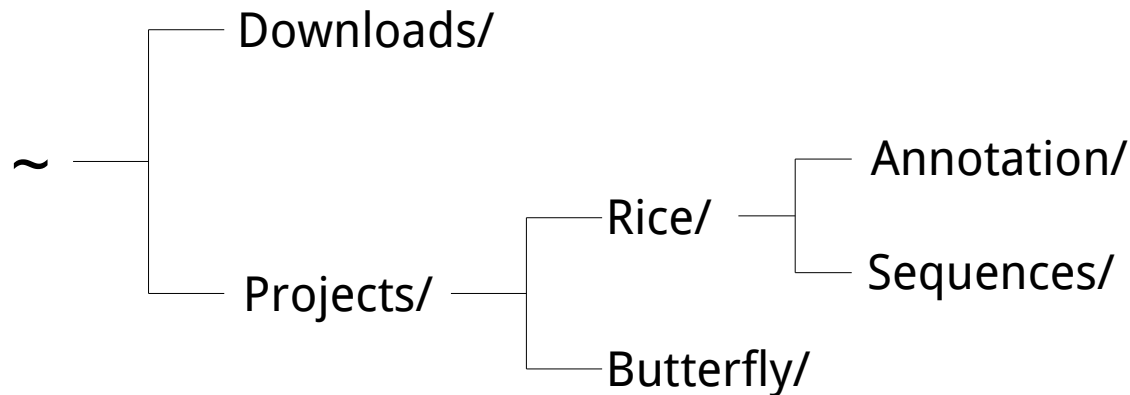
Moving data around

If we have following directory structure...



Moving data around

If we have following directory structure...



```
~ $ mkdir -p Projects/{Rice/{Annotation,Sequences},Butterfly}
~ $ cd ~/Downloads
~ $
```



Home

Training

Software Support

Data Mining Support

About BITS

Linux for Bioinformatics

DOWNLOAD TRAINING MATERIAL

- Lecture: "Dive into Linux" Introduction, distributions and installation - [Slides](#) by Joachim Jacob
- Lecture: "Dive into Linux" Software installation, graphical and command-line - [Slides](#) by Joachim Jacob
- Lecture: "Linux for Bioinformatics" Useful commands for bioinformatics - [Slides](#) by Joachim Jacob
- Data: [sample.sam](http://dl.dropbox.com/u/58174806/Linuxsample.sam)

Right click

LINKS

Quick Reference

- [Reference card](#)

Text tools tutorial

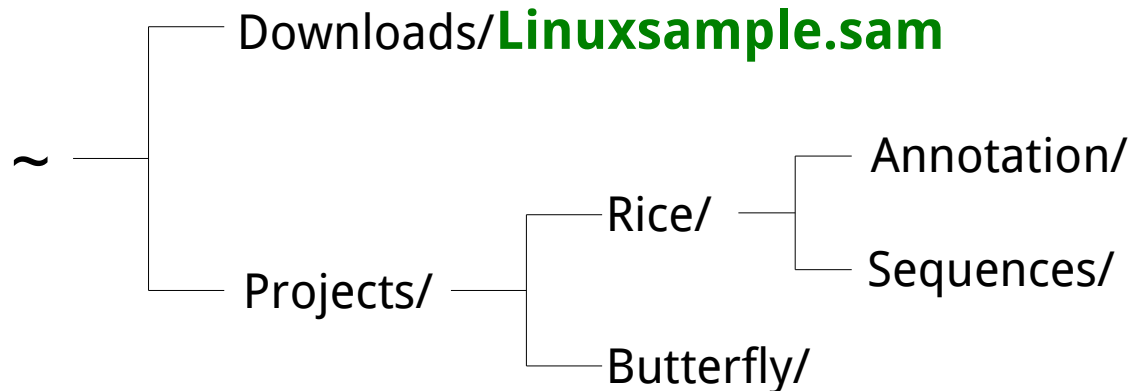
- [Linux' grep command](#)
- [Manual to AWK](#)
- [One-line scripts](#)

```
~ $ wget http://dl.dropbox.com/u/58174806/Linuxsample.sam
```

terfly}

Moving data around

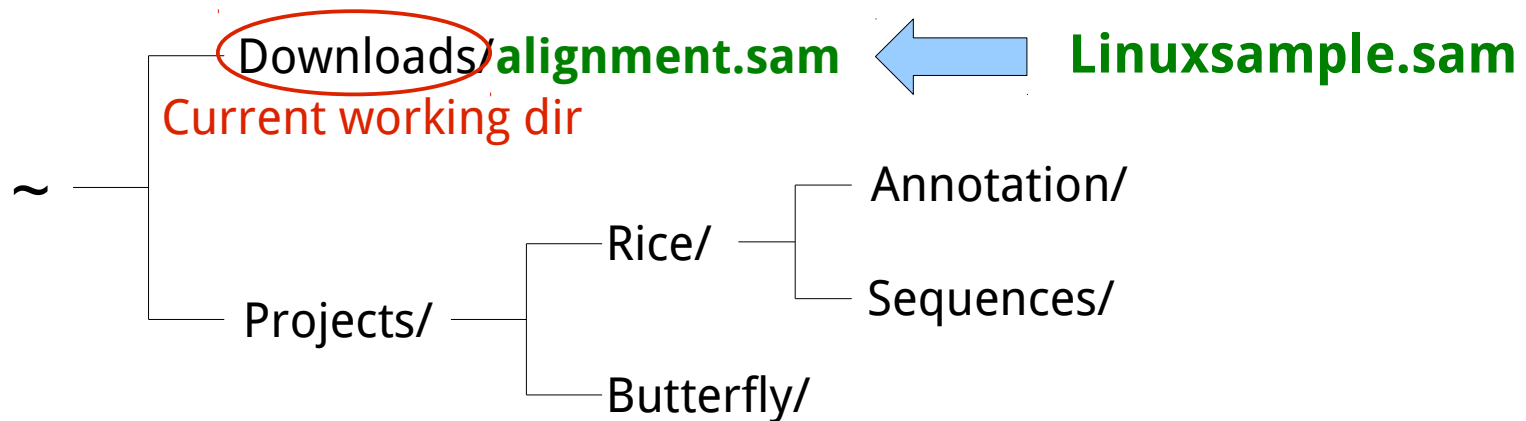
If we have following directory structure...



```
~ $ mkdir -p Projects/{Rice/{Annotation,Sequences},Butterfly}
~ $ cd ~/Downloads
~ $ wget http://dl.dropbox.com/u/58174806/Linuxsample.sam
```

Moving data around

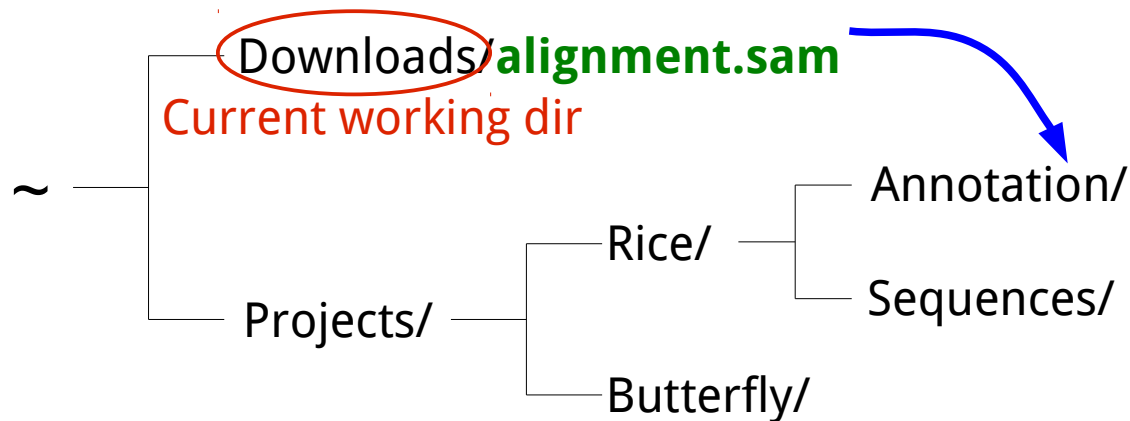
Rename the downloaded Linuxsample.sam



```
~/Downloads $ pwd
/home/joachim/Downloads
~/Downloads $ ls
Linuxsample.sam
~/Downloads $
```

Moving data around

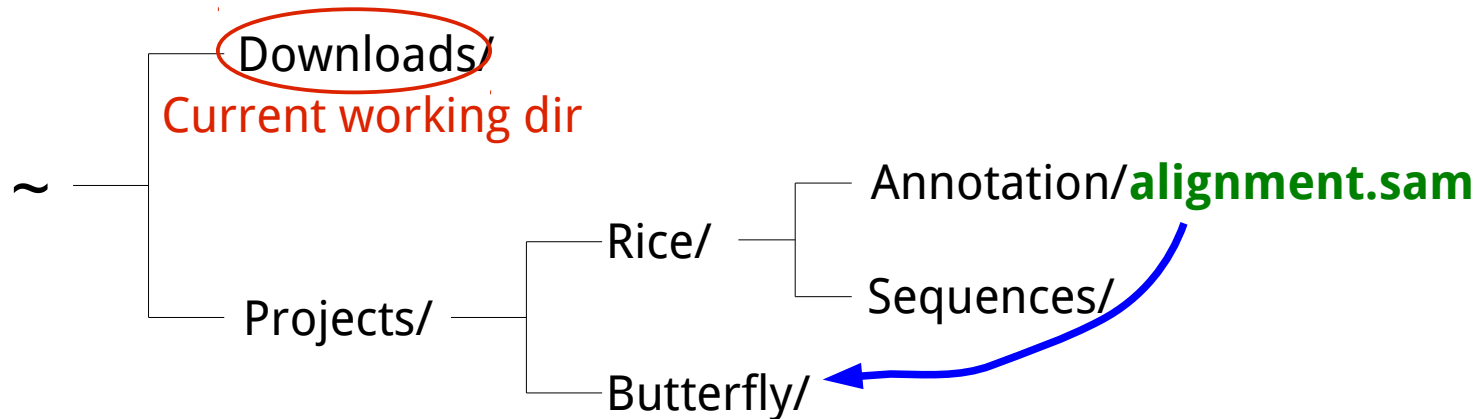
Move the sam file



```
~/Downloads $
```

Moving data around

Copy the sam file

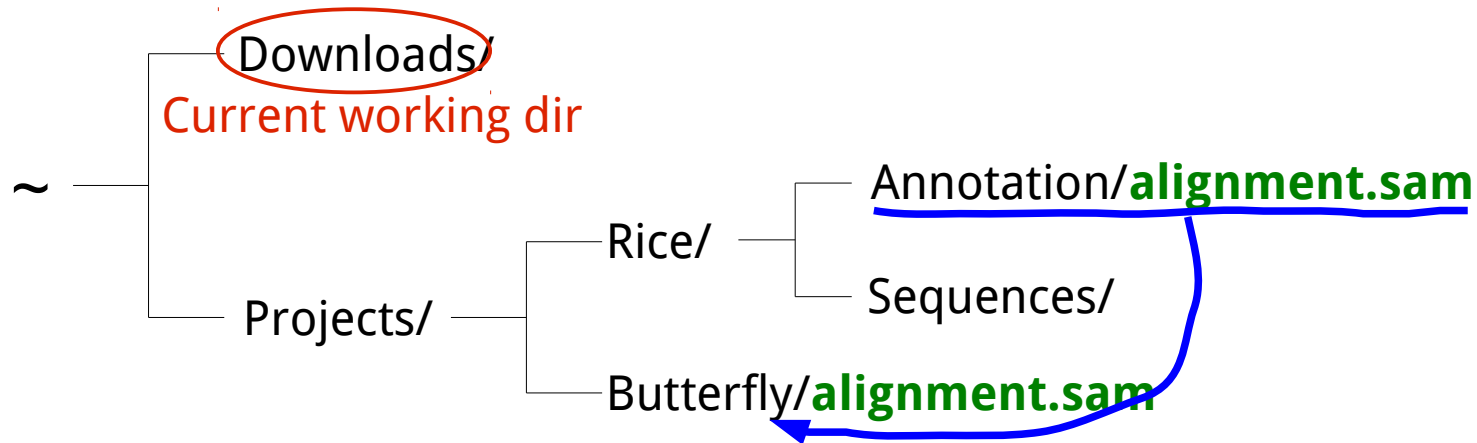


```
~/Downloads $ cp
```

```
\
```

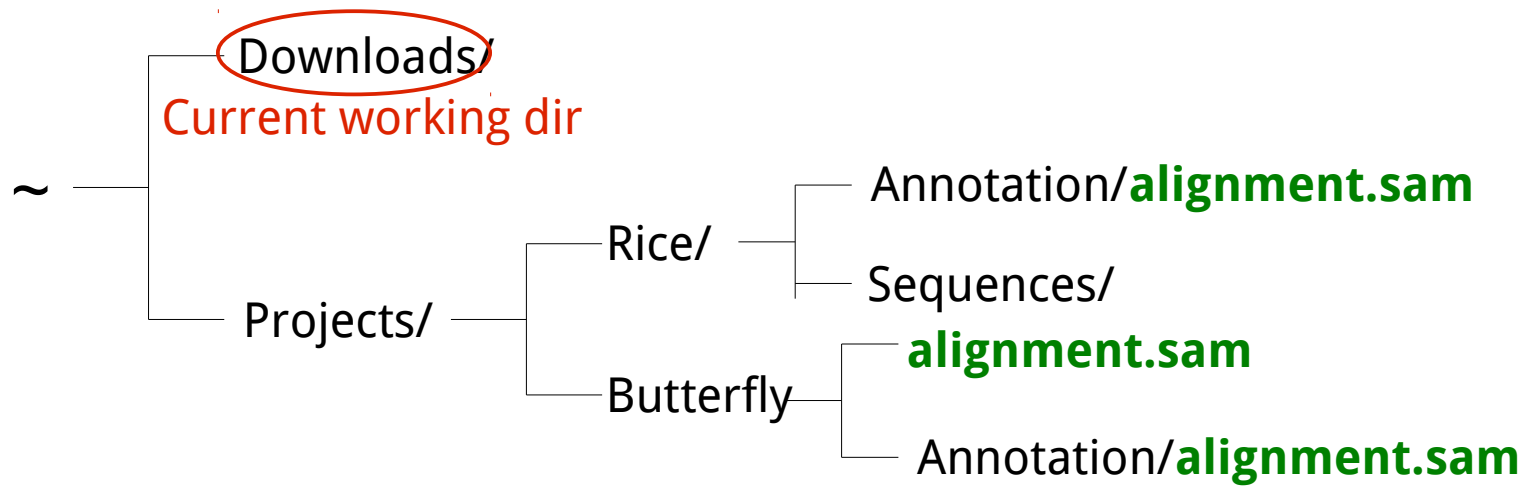

Moving data around

Copy the complete directory



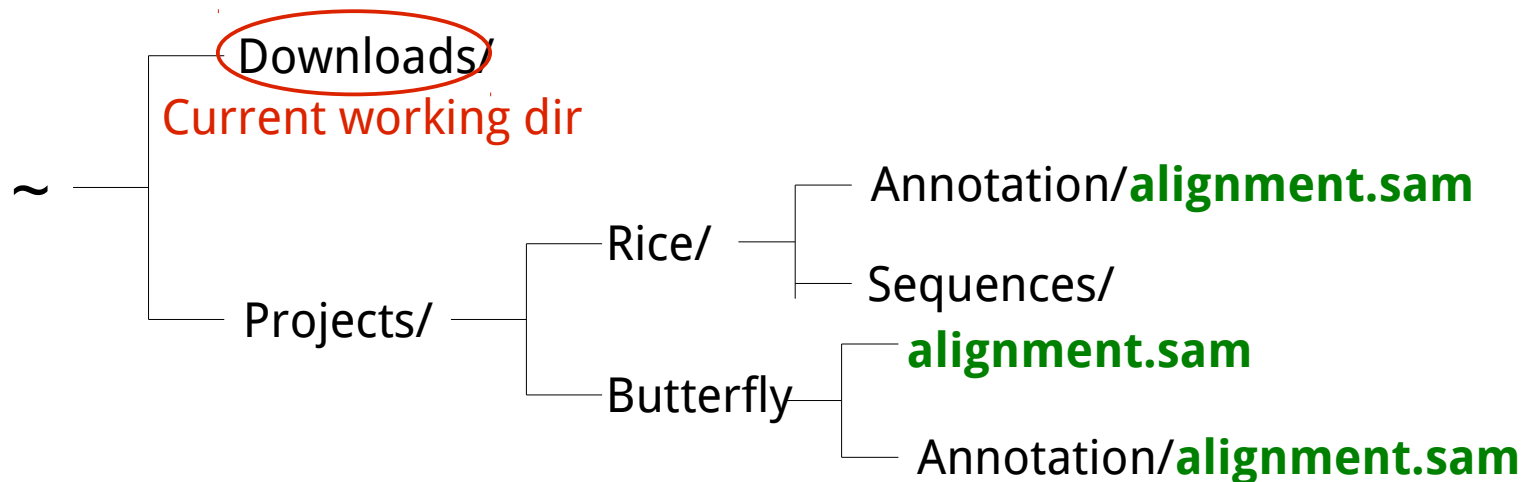
```
~/Downloads $ c
```

Reading files



What kind of file is .sam?

Reading files



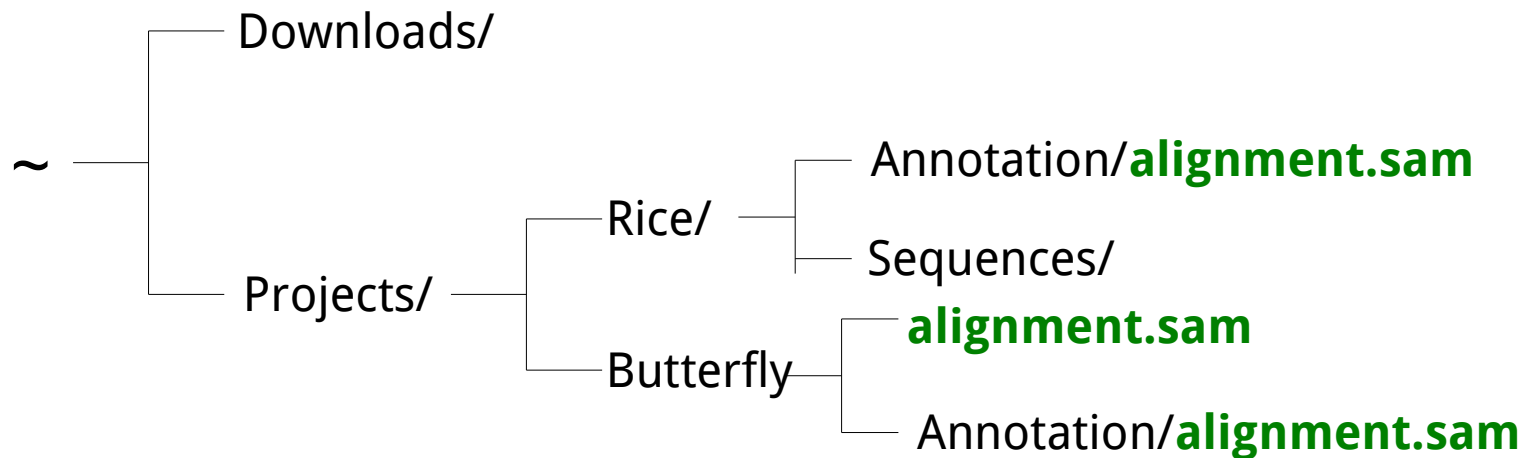
\$ **cat** : display the content of the file **at once**

\$ **less** : display the content **page by page**

\$ **nano** : **edit** the content of the file

Reading files

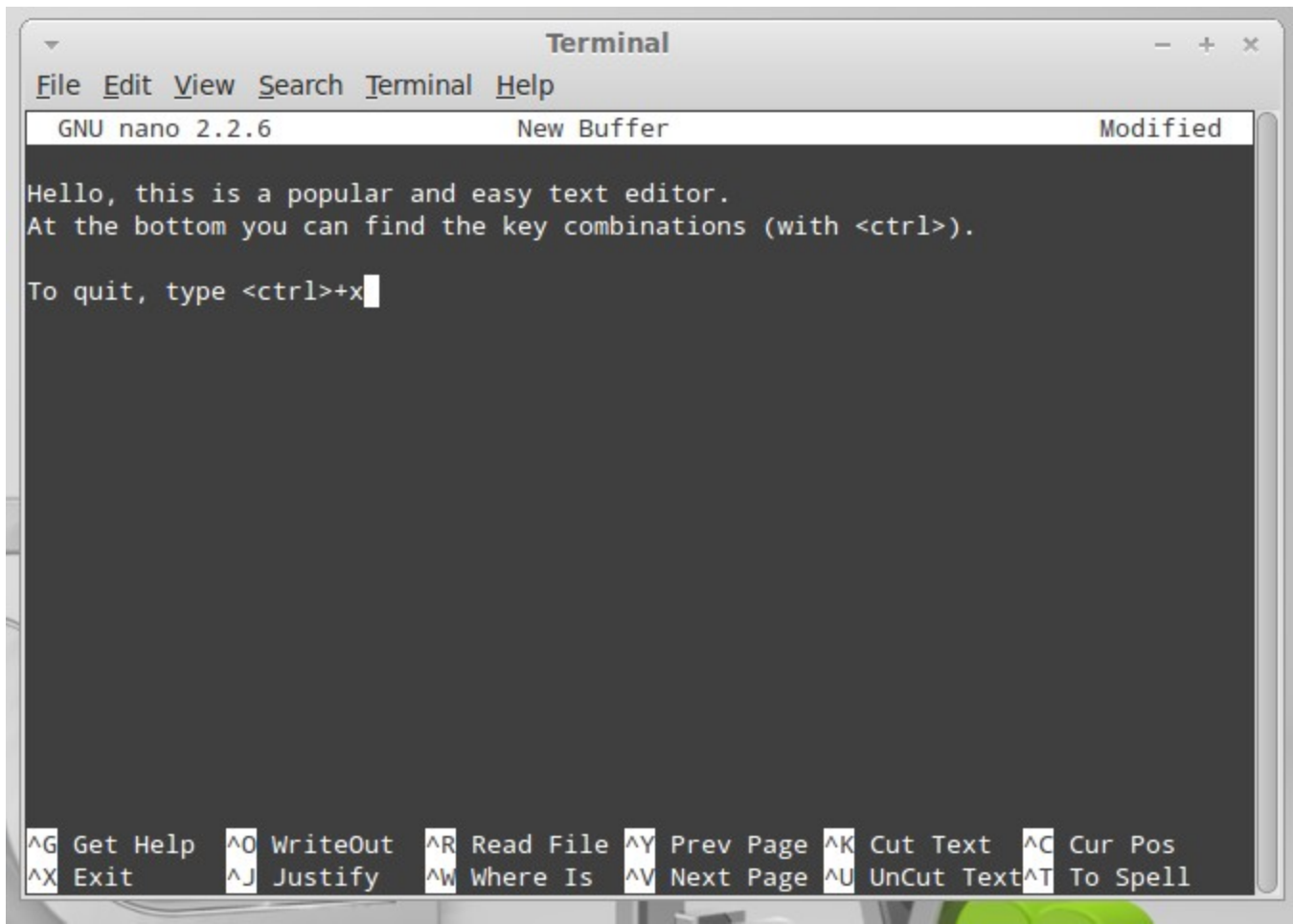
Try the 3 commands below



```
~/Projects/Butterfly $ cat alignment.sam
~/Projects/Butterfly $ less alignment.sam
~/Projects/Butterfly $ nano alignment.sam
```

Editing files

nano is a popular text editor to edit files.



The screenshot shows a terminal window titled "Terminal" with standard window controls. Inside, the GNU nano 2.2.6 text editor is open. The top status bar shows "GNU nano 2.2.6", "New Buffer", and "Modified". The editor's content area contains the following text:

```
Hello, this is a popular and easy text editor.  
At the bottom you can find the key combinations (with <ctrl>).  
To quit, type <ctrl>+x
```

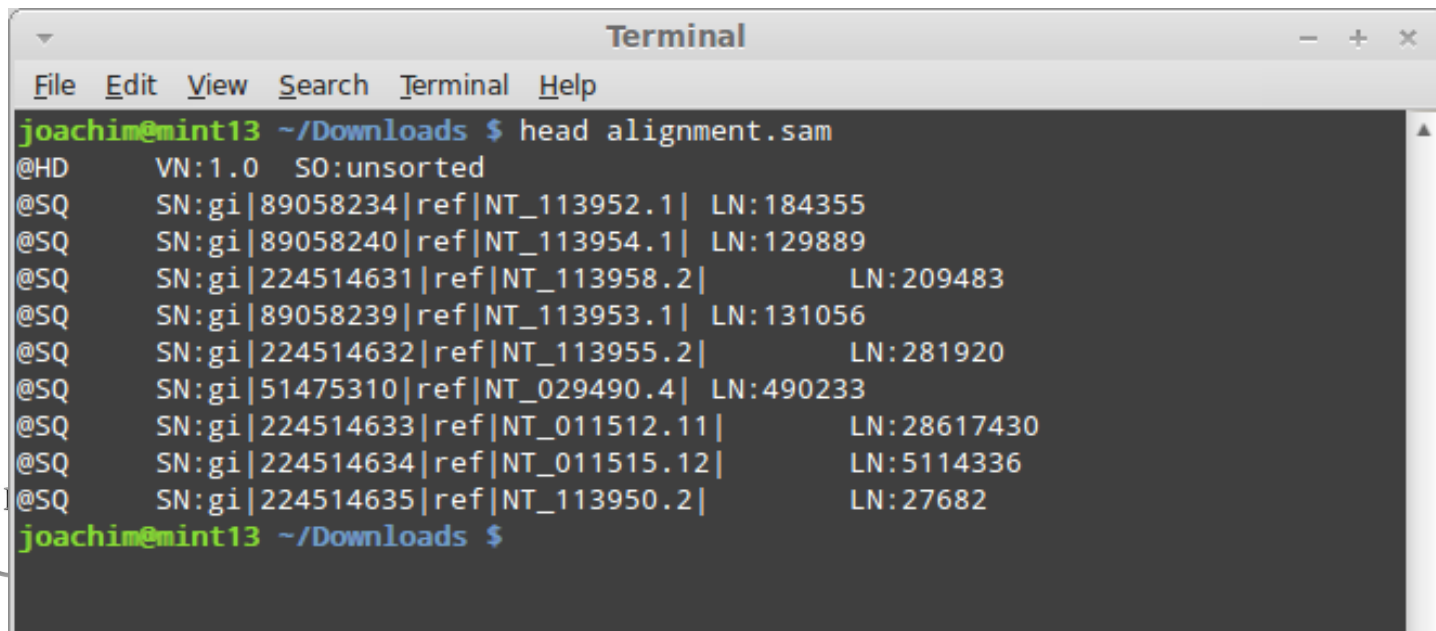
The cursor is positioned at the end of the third line. The bottom status bar displays a grid of keyboard shortcuts for various nano editor functions:

^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

Reading files

Display first or last lines of a text file, with **head** or **tail**.

```
~/Projects/Butterfly $ head alignment.sam  
~/Projects/Butterfly $ tail alignment.sam
```



```
Terminal  
File Edit View Search Terminal Help  
joachim@mint13 ~/Downloads $ head alignment.sam  
@HD      VN:1.0   SO:unsorted  
@SQ      SN:gi|89058234|ref|NT_113952.1|  LN:184355  
@SQ      SN:gi|89058240|ref|NT_113954.1|  LN:129889  
@SQ      SN:gi|224514631|ref|NT_113958.2|      LN:209483  
@SQ      SN:gi|89058239|ref|NT_113953.1|  LN:131056  
@SQ      SN:gi|224514632|ref|NT_113955.2|      LN:281920  
@SQ      SN:gi|51475310|ref|NT_029490.4|  LN:490233  
@SQ      SN:gi|224514633|ref|NT_011512.11|      LN:28617430  
@SQ      SN:gi|224514634|ref|NT_011515.12|      LN:5114336  
@SQ      SN:gi|224514635|ref|NT_113950.2|      LN:27682  
joachim@mint13 ~/Downloads $
```

Question

How can I display the first 20 lines of a text file?

```
~ $ head --help  
  
...  
  
-n, --lines=[-]K      print the first K lines instead of the first 10;  
                        with the leading `-', print all but the last  
                        K lines of each file  
  
...
```

Using the terminal efficiently

1. use arrow keys



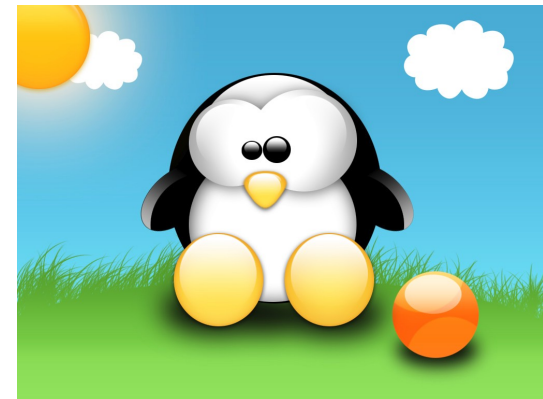
Using the terminal efficiently

- The program '**history**' keeps track of the last ~500 commands you have typed.

```
joachim@joachim-VirtualBox ~ $ history
 1 bowtie
 2 clear
 3 bowtie --version
 4 bowtie --version
 5 ./snap
 6 ./bowtie2
 7 sudo add-apt-repository
 8 sudo add-apt-repository ppa:webupd8team/tribler
 9 sudo apt-get update
10 sudo apt-get install tribler
11 tribbler
12 tribler
13 ls -l
14 ls -l
15 ls -l
16 ll
```

Using the terminal efficiently

- 1. use arrow keys*
- 2. use tab expansion*



Using the terminal efficiently

- Autocompletion, aka **tab expansion**: type the first letters of the program or file, and then press <tab> key.

- ```
$ cd /h<tab>
```

```
$ cd /home/
```

However

```
$ cd /b<tab>
```

 gives you audible feedback:

- there is no expansion possible
- there is more than one way to expand

```
$ cd /b<tab><tab>
```

 shows suitable expansions  

```
bin/ boot/
```

```
$ cd /bo<tab>
```

```
$ cd /boot/
```

# Using the terminal efficiently

- 1. use arrow keys*
- 2. use tab expansion*
- 3. use shorthand notations*



# Using the terminal efficiently

Use shorthand notations for common directories:

- `~` is your home directory
- `.` is the current directory
- `..` is the directory one level up

To execute a previous command `cmd` again you can use:

```
$!cmd
```

```
e.g. $!cd
```

# Exercise: getting large data files.



→ [Exercise link](#)

# Keywords

Root directory

path

home

terminal

Command line

user

bash

argument

recursively

command line options

# Break

