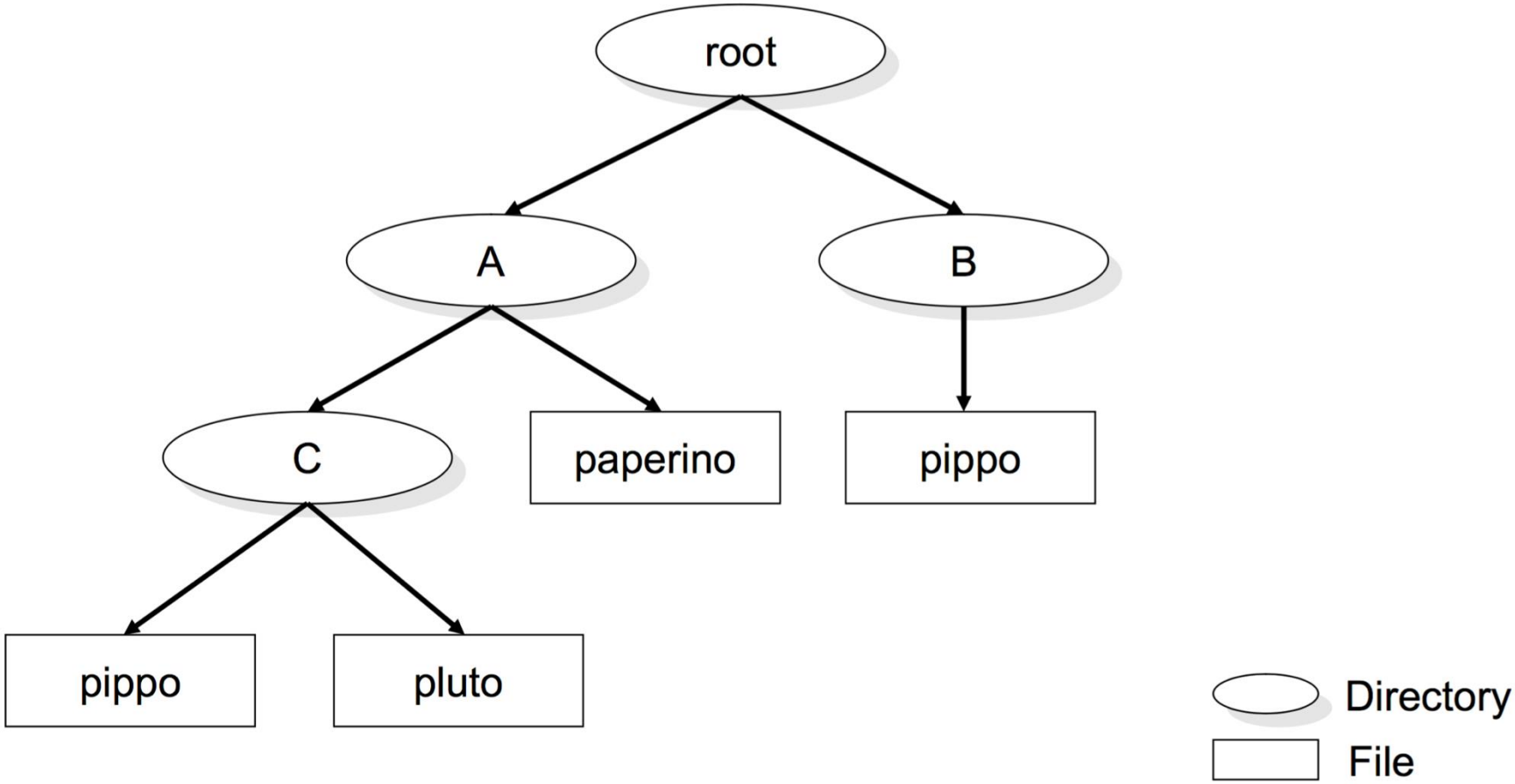
OPerating Systems Laboratory
OPS*Lab*

PART II
Bash Command Line

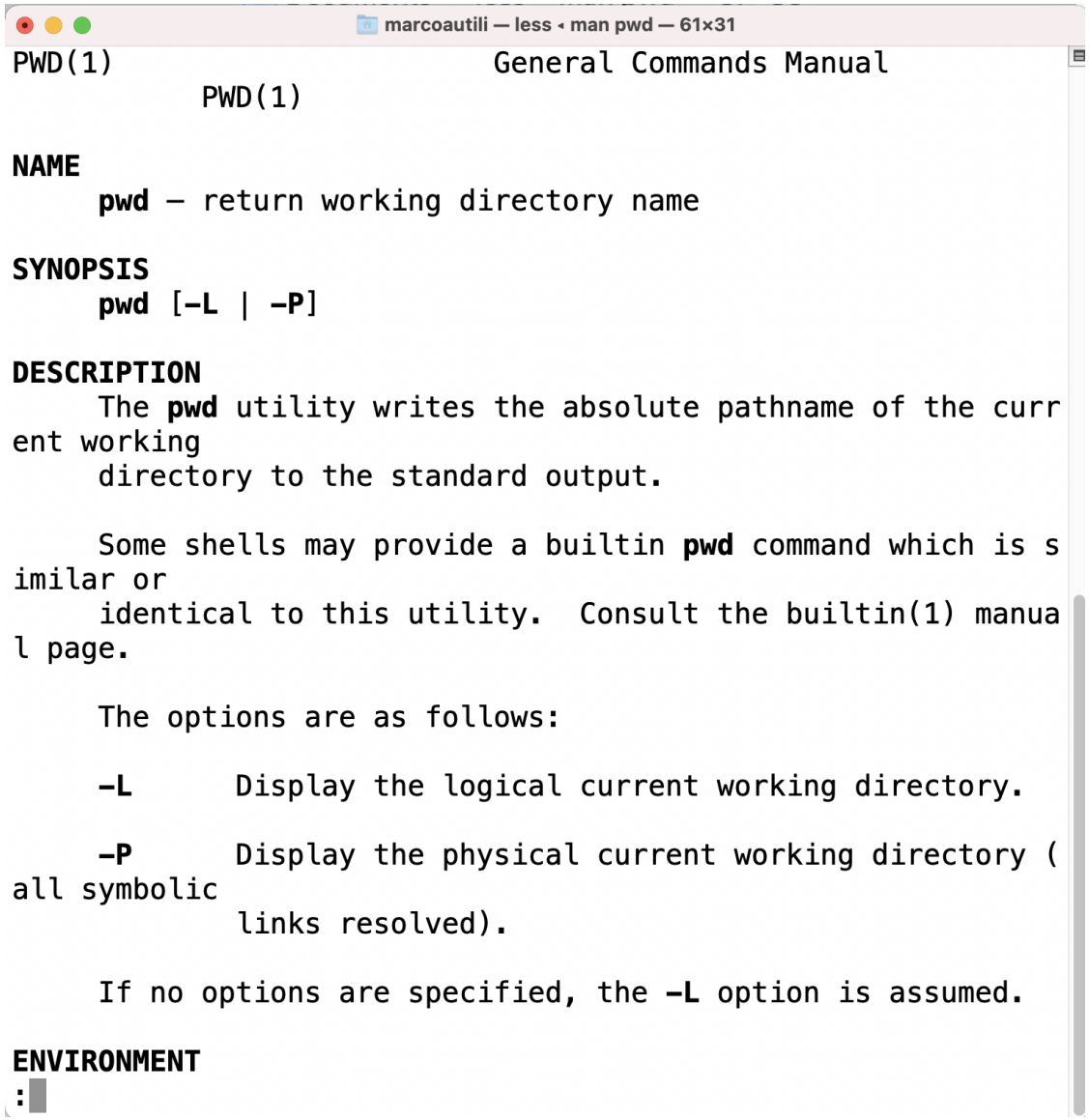
Prof. Marco Autili
University of L'Aquila
marco.autili@univaq.it
http://people.disim.univaq.it/marco.autili/

File System



Manual Pages

- man <command name>
  - man pwd
  - man 1 pwd
  - man s1 pwd
- press q to exit the manual pages



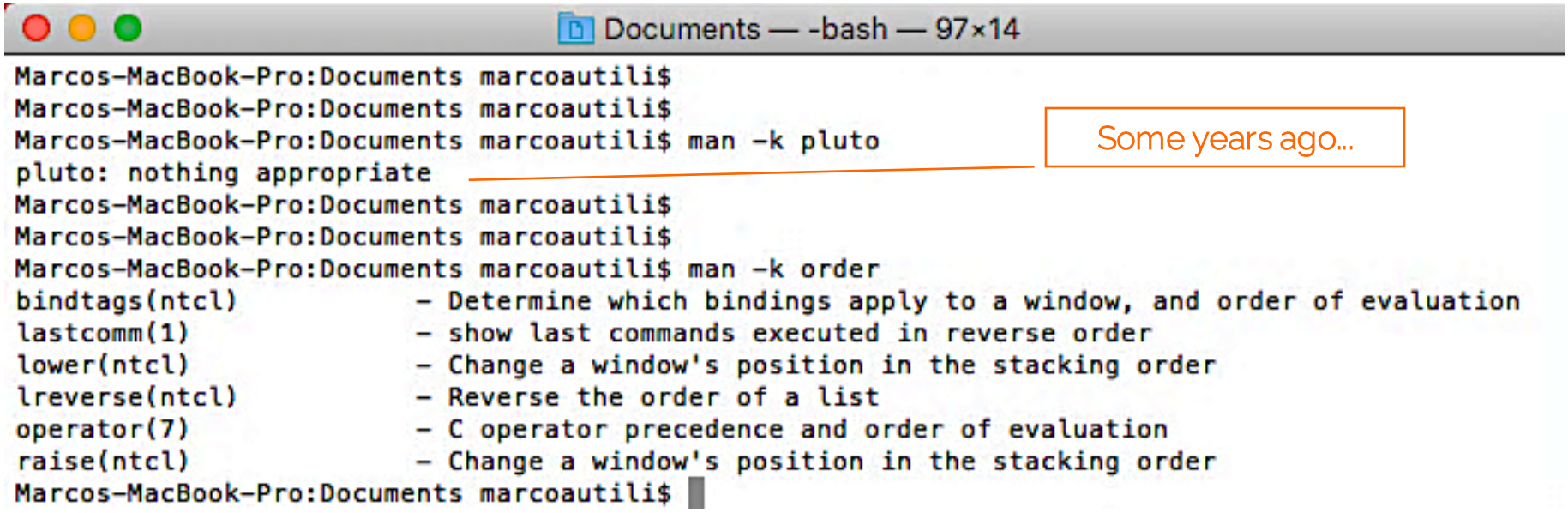
Searching Manual Pages

Search within all the manual pages

- man -k <search term>

Search within specific manual pages

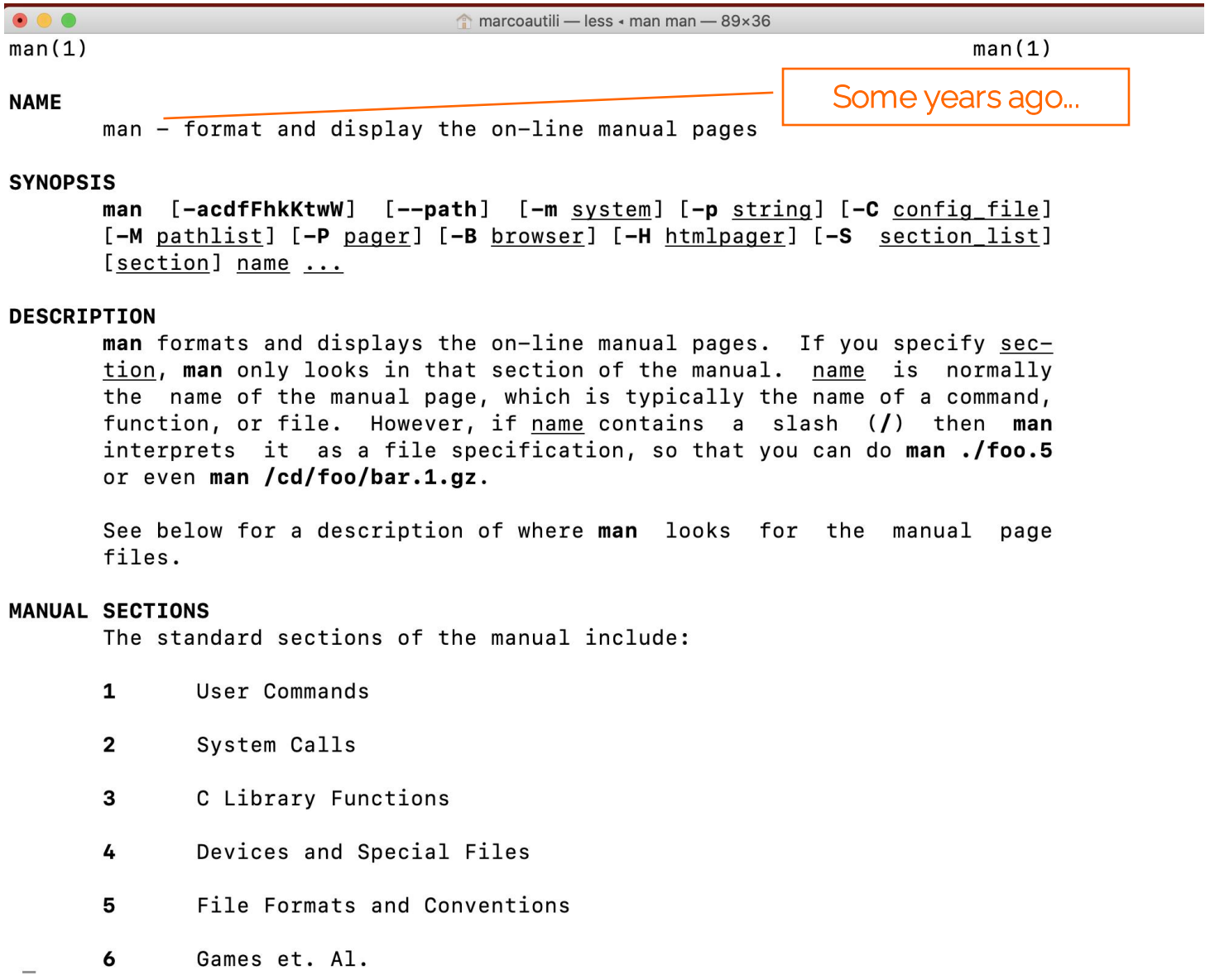
- while on the page, type forward slash '/' followed by the term you want to search for, then hit 'enter'
- press 'n' to go to the next occurrence



Want to know how man works?

Simply type

- man man



Want to know how man works?

Simply type

- man man

Often a wrapper for the man -k command

On MacOS Ventura in 2023…

marcoautili — less ‹ man man — 81×33

```
MAN(1)                    General Commands Manual                   MAN(1)

NAME
     man, apropos, whatis – display online manual documentation pages

SYNOPSIS
     man [-adho] [-t | -w] [-M manpath] [-P pager] [-S mar
         [-m arch[:machine]] [-p [eprtv]] [mansect] page ...

     man -f [-d] [-M manpath] [-P pager] [-S mansect] keyword ...
     whatis [-d] [-s mansect] keyword ...

     man -k [-d] [-M manpath] [-P pager] [-S mansect] keyword ...
     apropos [-d] [-s mansect] keyword ...

DESCRIPTION
     The man utility finds and displays online manual documentation pages.  If
     mansect is provided, man restricts the search to the specific section of
     the manual.

     The sections of the manual are:
           1.   General Commands Manual
           2.   System Calls Manual
           3.   Library Functions Manual
           4.   Kernel Interfaces Manual
           5.   File Formats Manual
           6.   Games Manual
           7.   Miscellaneous Information Manual
           8.   System Manager's Manual
           9.   Kernel Developer's Manual

     Options that man understands:
:
```

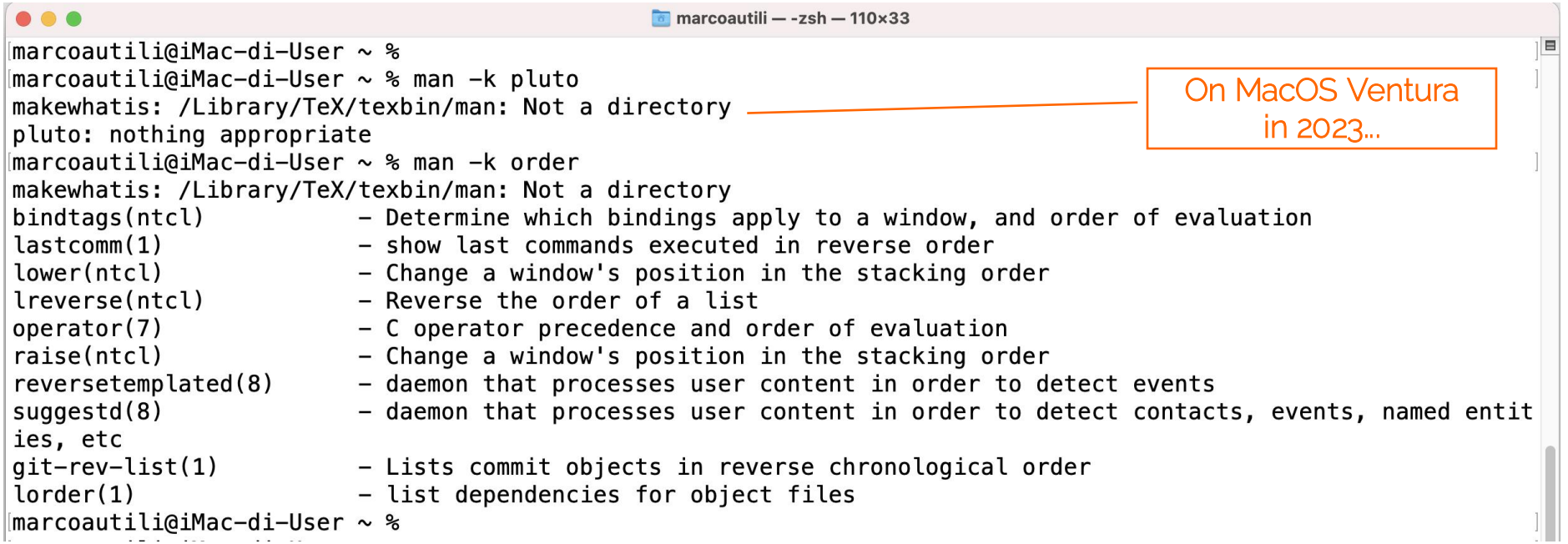
Searching Manual Pages

Search within all the manual pages

- man -k <search term>

Search within specific manual pages

- while on the page, type forward slash '/' followed by the term you want to search for, then hit 'enter'
- press 'n' to go to the next occurrence

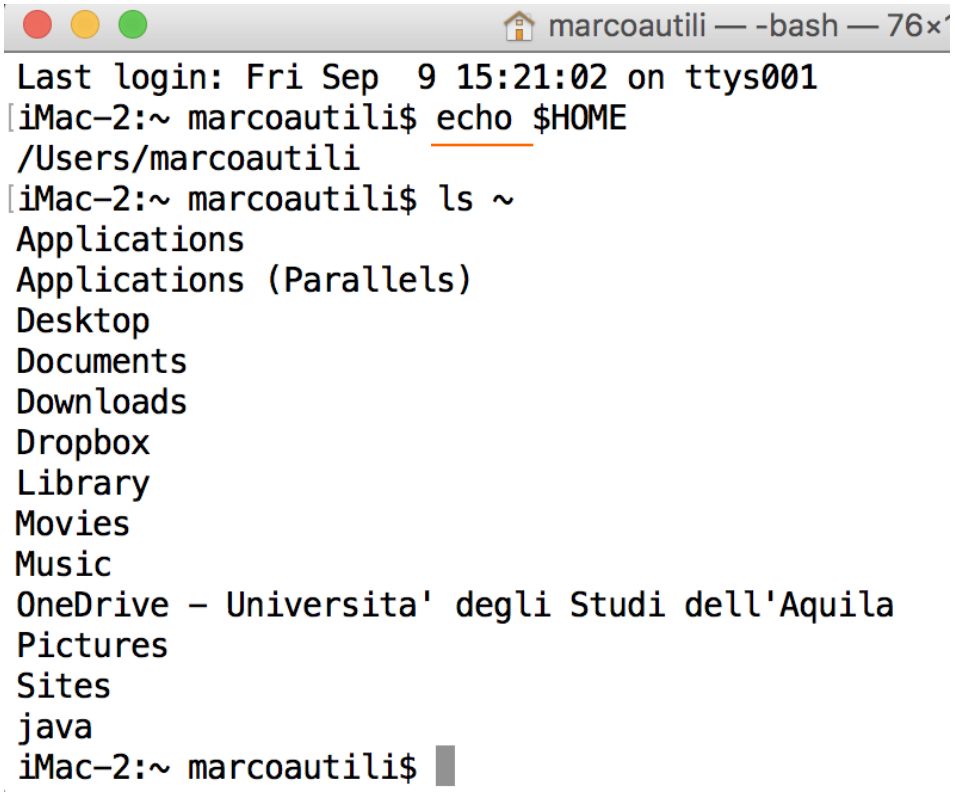


More on apropos

- https://en.wikipedia.org/wiki/Apropos_(Unix)

Path shortcuts

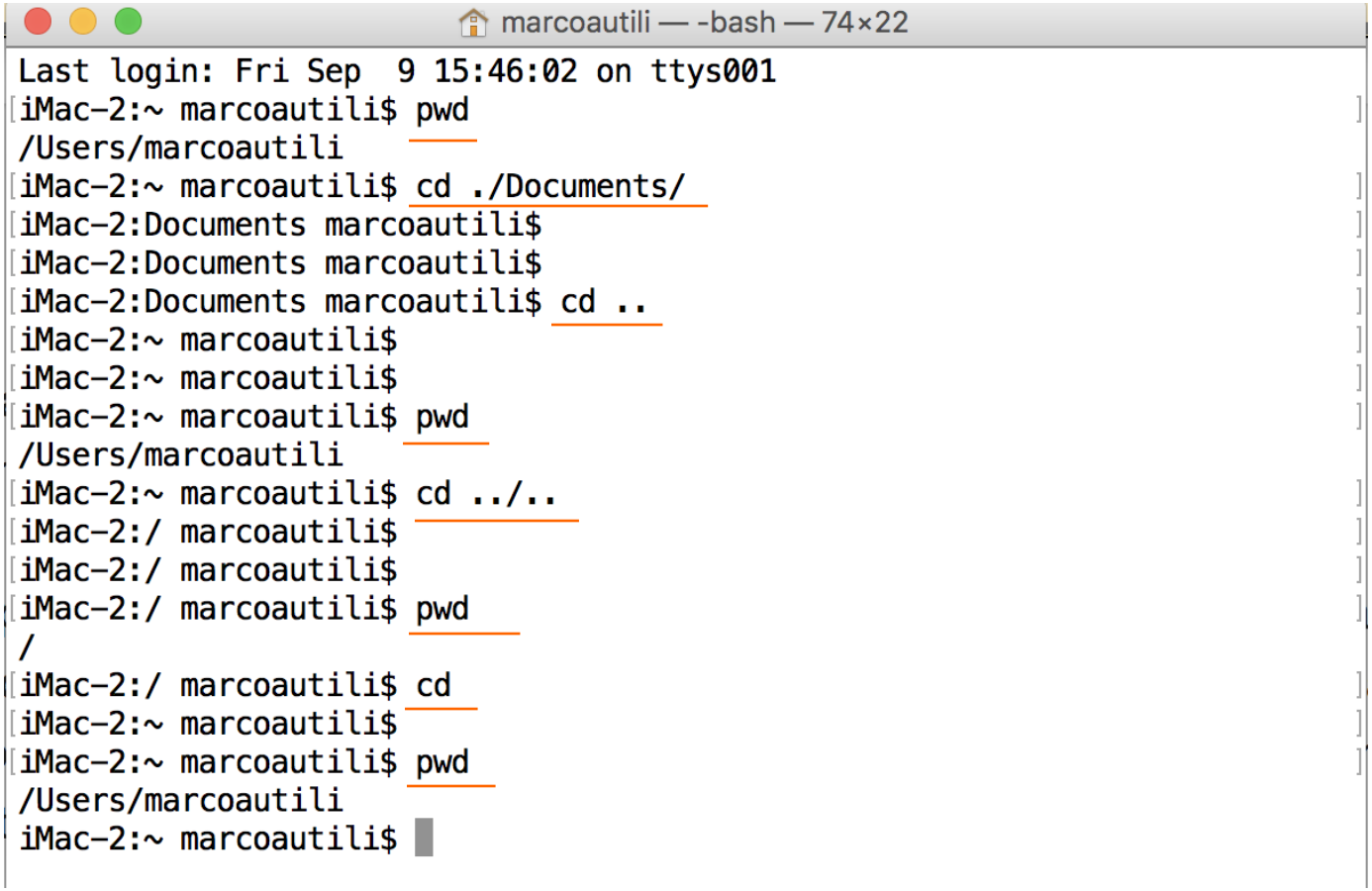
- ~ (tilde) - shortcut for the home directory
  - e.g., /Users/marcoautili/Documents
    is equivalent to ~/Documents
  - try echo ~
- . (dot) - reference to the current directory
- .. (dotdot) - reference to the parent directory (it can use this several times, e.g., ls ../../)



Navigating the file system

cd [location]

- it takes you back to your home directory when called without any arguments



Listing directory contents

- Remember Absolute and Relative Paths

- pwd (stands for Print Working Directory)

- ls [options] [location] (stands for list)
  - ls (listing of the current directory)
  - ls -l (long listing of the current directory)
  - ls /etc (listing of the /etc directory in the root /)
  - ls -l /etc (long listing of the /etc directory)
  - ls documents or ls ./documents (listing of the documents directory in the current directory)
  - ls -la (include directory entries whose names begin with a dot "." - i.e., hidden files or directories, see next slide)
  - ls -1 (list one entry per line)
  - ls -1t (list one entry per line and sort by time modified - most recently modified first- before sorting the operands by lexicographical order)
  - see also ls -all and compare it with ls -al and ls -la

→| Tab Completion helps us typing command by offering the shell's quoting and escape mechanism

Hidden files and directories

A file or directory whose name begins with a "." (full stop) is considered to be hidden

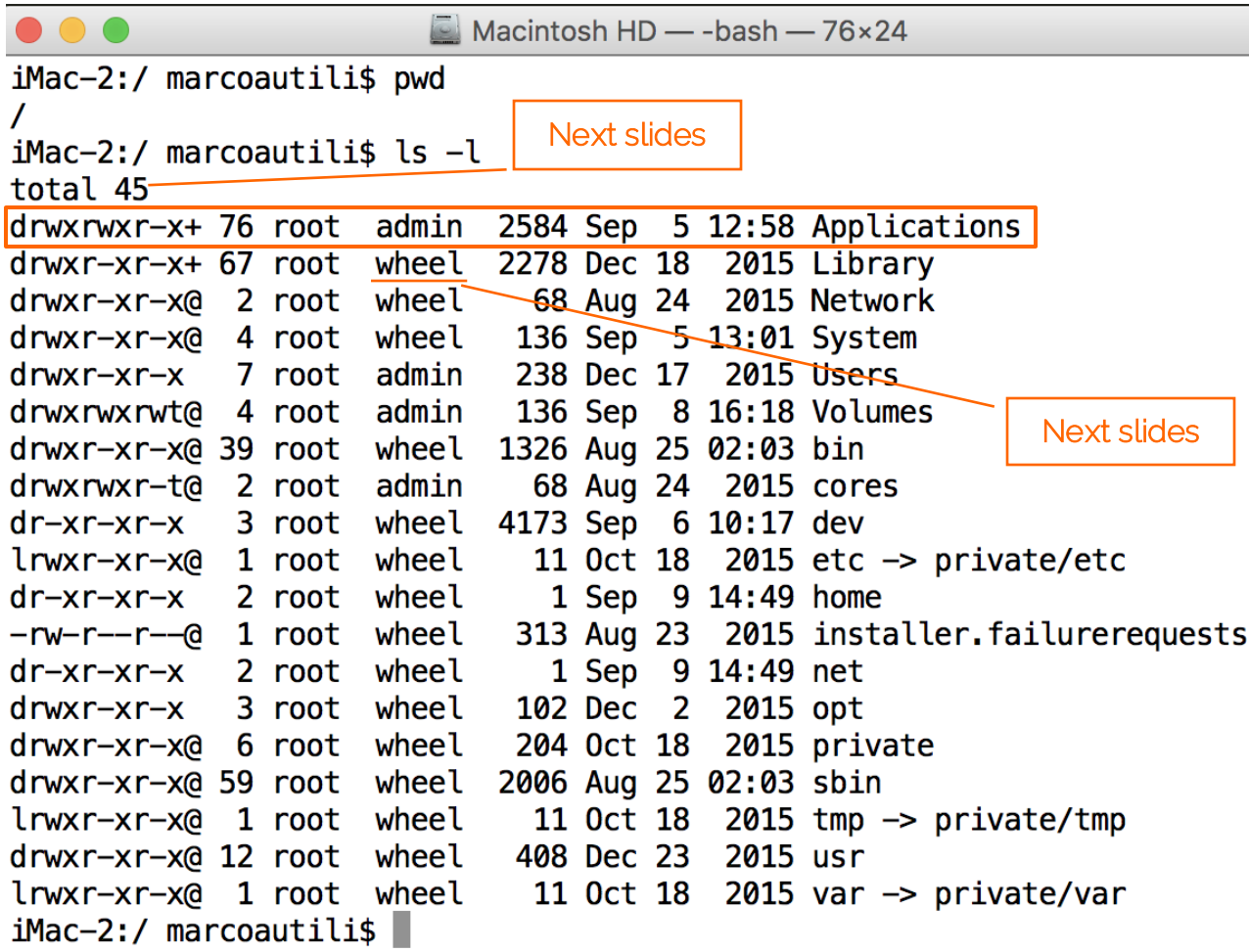
Documents — -zsh — 86×33

```
marcoautili@iMac-di-User ~ %
marcoautili@iMac-di-User ~ %
marcoautili@iMac-di-User ~ %
marcoautili@iMac-di-User ~ % cd Documents
marcoautili@iMac-di-User Documents %
marcoautili@iMac-di-User Documents %
marcoautili@iMac-di-User Documents % ls -l
total 3744
drwxr-xr-x  3 marcoautili  staff       96 Sep 10  2018 Parallels
drwxr-xr-x  3 marcoautili  staff       96 Jul 28  2015 RDC Connections
-rw-r--r--@ 1 marcoautili  staff     7424 Dec 12  2018 Terminal Saved Output.txt
-rw-r--r--@ 1 marcoautili  staff      178 May 23  2015 Untitled.tex.rtf
drwxr-xr-x@ 2 marcoautili  staff       64 Oct 28  2019 WebEx
-rw-r--r--@ 1 marcoautili  staff      402 Mar 14  2018 desktop.ini
-rw-r--r--@ 1 marcoautili  staff  1899478 Dec  5  2019 website.zip
drwxr-xr-x  5 marcoautili  staff      160 Sep 10  2018 workspace
marcoautili@iMac-di-User Documents %
marcoautili@iMac-di-User Documents % ls -la
total 3760
drwx------+ 13 marcoautili  staff      416 Dec  5  2019 .
drwxr-xr-x+ 56 marcoautili  staff     1792 Oct  5 16:30 ..
-rw-r--r--@  1 marcoautili  staff     6148 Nov 12  2017 .DS_Store
-rw-------   1 marcoautili  staff        0 Sep 10  2018 .localized
drwxr-xr-x   2 marcoautili  staff       64 Oct 30  2015 .parallels-vm-directory
drwxr-xr-x   3 marcoautili  staff       96 Sep 10  2018 Parallels
drwxr-xr-x   3 marcoautili  staff       96 Jul 28  2015 RDC Connections
-rw-r--r--@  1 marcoautili  staff     7424 Dec 12  2018 Terminal Saved Output.txt
-rw-r--r--@  1 marcoautili  staff      178 May 23  2015 Untitled.tex.rtf
drwxr-xr-x@  2 marcoautili  staff       64 Oct 28  2019 WebEx
-rw-r--r--@  1 marcoautili  staff      402 Mar 14  2018 desktop.ini
-rw-r--r--@  1 marcoautili  staff  1899478 Dec  5  2019 website.zip
drwxr-xr-x   5 marcoautili  staff      160 Sep 10  2018 workspace
marcoautili@iMac-di-User Documents %
```

Fields displayed by the ls command

- First character – e.g., file (-), directory (d), soft link (l), etc.

  (soft links are also called symbolic links)
- Next 9 characters - permissions
- The next field - number of hard links of the file, when referring to a file, or the number of contained directory entries, when referring to a directory (more on that later on)
- The next field - owner name (root in this case)
- The next field - owner group (e.g., admin)
- The next field - file size
- The next field - file modification time
- The last field - name of the file or directory or link



File permissions (three sets of characters, three times, indicating permissions for owner, group and other

r = readable w = writable x = executable

Permissions for a directory

- Note that read permission for a directory and execute permission for a directory mean different things
- Read permission lets us read the directory, obtaining a list of all the filenames in the directory
- Execute permission lets us pass through the directory when it is a component of a pathname that we are trying to access
- We need to search the directory to look for a specific filename

File permissions (three sets of characters, three times, indicating permissions for owner, group and other

r = readable w = writable x = executable

The wheel group

- In computing, the term wheel refers to a user account with a wheel bit, a system setting that provides additional special system privileges that empower a user to execute restricted commands that ordinary user accounts cannot access.
- The term is derived from the slang phrase "big wheel", referring to a person with great power or influence.
- It was first used in this context with regard to the TENEX operating system, later distributed under the name TOPS-20 in the 1960s and early 1970s.
- The term was adopted by Unix users in the 1980s, due to the movement of operating system developers and users from TENEX/TOPS-20 to Unix.
- Modern Unix implementations generally include a security protocol that requires a user to be a member of the wheel user privileges group in order to gain superuser access to a machine by using the su command.
- Modern Unix systems use user groups to control access privileges. The wheel group is a special user group used on some Unix systems to control access to the su command, which allows a user to masquerade as another user (usually the super user)

An anticipation on soft links

With a symbolic link, the actual contents of the file (i.e., the data blocks) store the name of the file that the symbolic link points to

For example, the filename in the directory entry is the three-character string var and the 11 bytes of data in the file are private/var (one byte for char)

More on that on PART 4

Macintosh HD — -bash — 76×24

```
iMac-2:/ marcoautili$ pwd
/
iMac-2:/ marcoautili$ ls -l
total 45
drwxrwxr-x+ 76 root  admin  2584 Sep  5 12:58 Applications
drwxr-xr-x+ 67 root  wheel  2278 Dec 18  2015 Library
drwxr-xr-x@  2 root  wheel    68 Aug 24  2015 Network
drwxr-xr-x@  4 root  wheel   136 Sep  5 13:01 System
drwxr-xr-x   7 root  admin   238 Dec 17  2015 Users
drwxrwxrwt@  4 root  admin   136 Sep  8 16:18 Volumes
drwxr-xr-x@ 39 root  wheel  1326 Aug 25 02:03 bin
drwxrwxr-t@  2 root  admin    68 Aug 24  2015 cores
dr-xr-xr-x   3 root  wheel  4173 Sep  6 10:17 dev
lrwxr-xr-x@  1 root  wheel    11 Oct 18  2015 etc -> private/etc
dr-xr-xr-x   2 root  wheel     1 Sep  9 14:49 home
-rw-r--r--@  1 root  wheel   313 Aug 23  2015 installer.failurerequests
dr-xr-xr-x   2 root  wheel     1 Sep  9 14:49 net
drwxr-xr-x   3 root  wheel   102 Dec  2  2015 opt
drwxr-xr-x@  6 root  wheel   204 Oct 18  2015 private
drwxr-xr-x@ 59 root  wheel  2006 Aug 25 02:03 sbin
lrwxr-xr-x@  1 root  wheel    11 Oct 18  2015 tmp -> private/tmp
drwxr-xr-x@ 12 root  wheel   408 Dec 23  2015 usr
lrwxr-xr-x@  1 root  wheel    11 Oct 18  2015 var -> private/var
iMac-2:/ marcoautili$
```

Macintosh HD — -bash — 72×7

```
drwxr-xr-x@   9 root  wheel   288 Oct 25 18:22 usr
lrwxr-xr-x@   1 root  wheel    11 Nov 11 19:07 var -> private/var
valintina-vaio:/ marcoautili$
valintina-vaio:/ marcoautili$
valintina-vaio:/ marcoautili$ readlink var
private/var
valintina-vaio:/ marcoautili$
```

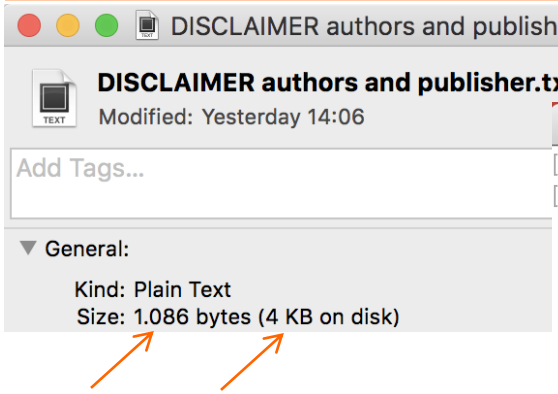
Filesystem block

The block size specifies the size that the filesystem uses to read and write data

- Larger block sizes will help improve disk I/O performance when using large files, such as databases. This happens because the disk can read or write data for a longer period of time before having to search for the next block.
- On the downside, if you are going to have a lot of smaller files on that filesystem, like the /etc, there the potential for a lot of wasted disk space.
- For example, if you set your block size to 4096, or 4K, and you create a file that is 256 bytes in size, it will still consume 4K of space on your hard drive. For one file that may seem trivial, but when your filesystem contains hundreds or thousands of files, this can add up.
- Block size can also affect the maximum supported file size on some filesystems.
    - This is because many modern filesystem are limited not by block size or file size, but by the number of blocks. Therefore, you would be using a "block size * max # of blocks = max file size" formula.

* http://www.tldp.org/LDP/sag/html/filesystems.html

Back to the size of files...



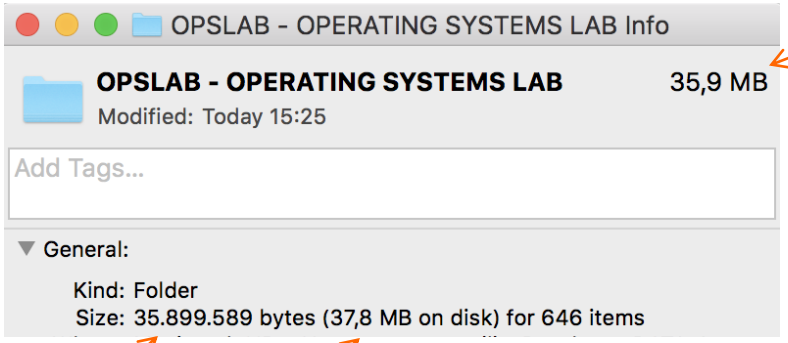
Reports allocation space on disk (and not absolute file space) as multiples of 512 bytes

Reports allocation space on disk (and not absolute file space) as multiples of 1024 bytes

Reports allocation space on disk (and not absolute file space) in human-readable format (Byte, Kilobyte, Megabyte, Gigabyte, Terabyte and Petabyte)

```
src — -bash — 91×32
iMac-2:src marcoautili$
iMac-2:src marcoautili$ ls -la
total 104
drwxr-xr-x@ 16 marcoautili  staff   544 Sep  8 16:52 .
drwxr-xr-x@  5 marcoautili  staff   170 Sep  9 17:35 ..
-rw-r--r--@  1 marcoautili  staff  6148 Sep  9 17:57 .DS_Store
-rw-r--r--@  1 marcoautili  staff  1086 Sep  8 14:06 DISCLAIMER authors and publisher.txt
-r--r--r--@  1 marcoautili  staff   656 Apr 27  2013 DISCLAIMER teacher.txt
-rw-r--r--@  1 marcoautili  staff   528 Apr 27  2013 Make.defines.freebsd
-rw-r--r--@  1 marcoautili  staff   526 Jul 22  2012 Make.defines.linux
-rw-r--r--@  1 marcoautili  staff   533 Sep  8 17:00 Make.defines.macos
-rw-r--r--@  1 marcoautili  staff   575 Apr 27  2013 Make.defines.solaris
-rw-r--r--@  1 marcoautili  staff    41 Apr 27  2013 Make.libapue.inc
-rw-r--r--@  1 marcoautili  staff   352 Sep  8 16:53 Makefile
-rw-r--r--@  1 marcoautili  staff   224 Sep  8 14:12 README.txt
drwxr-xr-x@ 12 marcoautili  staff   408 Sep  8 17:29 chap1-intro
drwxr-xr-x@  3 marcoautili  staff   102 Mar 20  2014 include
drwxr-xr-x@ 38 marcoautili  staff  1292 Sep  8 17:29 lib
-r-xr-xr-x@  1 marcoautili  staff   467 Jul  5  2012 systype.sh
iMac-2:src marcoautili$
iMac-2:src marcoautili$
iMac-2:src marcoautili$ du DISCLAIMER\ authors\ and\ publisher.txt
8       DISCLAIMER authors and publisher.txt
iMac-2:src marcoautili$
iMac-2:src marcoautili$
iMac-2:src marcoautili$ du -k DISCLAIMER\ authors\ and\ publisher.txt
4       DISCLAIMER authors and publisher.txt
iMac-2:src marcoautili$
iMac-2:src marcoautili$
iMac-2:src marcoautili$ du -h DISCLAIMER\ authors\ and\ publisher.txt
4.0K    DISCLAIMER authors and publisher.txt
iMac-2:src marcoautili$
```

Back to the size of files...



This is the size of space that is used to store the meta information (i.e., the directory entries) for the current directory

A directory is just a file that contains directory entries (back on this later)

```
OPSLAB - OPERATING SYSTEMS LAB — -bash — 92×16
iMac-2:OPSLAB - OPERATING SYSTEMS LAB marcoautili$
iMac-2:OPSLAB - OPERATING SYSTEMS LAB marcoautili$ ls -la
total 480
drwxr-xr-x@ 10 marcoautili  staff     340 Sep  9 15:25 .
drwxr-xr-x@  8 marcoautili  staff     272 Sep  8 22:49 ..
-rw-r--r--@  1 marcoautili  staff    8196 Sep  9 19:21 .DS_Store
drwxr-xr-x@  5 marcoautili  staff     170 Sep  7 13:46 EXAMS
drwxr-xr-x@  4 marcoautili  staff     136 Sep  9 15:01 LEZIONI
drwxr-xr-x@  5 marcoautili  staff     170 Sep  9 17:35 OPSLab-teaching-material-private
drwxr-xr-x@ 11 marcoautili  staff     374 Sep  9 17:57 REFERENCE TEXTBOOKs
-rw-r--r--@  1 marcoautili  staff  228550 Sep  7 13:22 SCHEDA-TUNING.pdf
drwxr-xr-x@  5 marcoautili  staff     170 Sep  7 13:50 SLIDES
-rw-r--r--@  1 marcoautili  staff      39 Jul 15 10:29 USEFUL LINKs.txt
iMac-2:OPSLAB - OPERATING SYSTEMS LAB marcoautili$ du -sh .
 36M    .
iMac-2:OPSLAB - OPERATING SYSTEMS LAB marcoautili$
```

That's the size of the directory content!

Total

Macintosh HD — -bash — 76×24

```
iMac-2:/ marcoautili$ pwd
/
iMac-2:/ marcoautili$ ls -l
total 45
drwxrwxr-x+ 76 root  admin  2584 Sep  5 12:58 Applications
drwxr-xr-x+ 67 root  wheel  2278 Dec 18  2015 Library
drwxr-xr-x@  2 root  wheel    68 Aug 24  2015 Network
drwxr-xr-x@  4 root  wheel   136 Sep  5 13:01 System
drwxr-xr-x   7 root  admin   238 Dec 17  2015 Users
drwxrwxrwt@  4 root  admin   136 Sep  8 16:18 Volumes
drwxr-xr-x@ 39 root  wheel  1326 Aug 25 02:03 bin
```

marcoautili — less ◂ man ls — 115×37

More on that later

```
The Long Format
  If the -l option is given, the following information is displayed for each file: file mode, number
  of links, owner name, group name, number of bytes in the file, abbreviated month, day-of-month
  file was last modified, hour file last modified, minute file last modified, and the pathname.  In
  addition, for each directory whose contents are displayed, the total number of 512-byte blocks
  used by the files in the directory is displayed on a line by itself, immediately before the infor-
  mation for the files in the directory.  If the file or directory has extended attributes, the per-
  missions field printed by the -l option is followed by a '@' character.  Otherwise, if the file or
  directory has extended security information (such as an access control list), the permissions
  field printed by the -l option is followed by a '+' character.  If the -% option is given, a '%'
  character follows the permissions field for dataless files and directories, possibly replacing the
  '@' or '+' character.
```

Total

If the output is to a terminal, the total number of blocks for all the files is output on a line before the listing

- ls -s Display the number of file system blocks actually used by each file, in units of 512 bytes, where partial units are rounded up to the next integer value
- The environment variable BLOCKSIZE (when set) overrides the unit size of 512 bytes

N.B.: Only files are counted (not directories)

Multiples of 8 are used since the read and write unit in my system is 4096
(i.e., 8 * 512 = 4096 bytes)

```
marcoautili — -bash — 85×26
iMac-di-User:~ marcoautili$ ls -ls
total 24
 0 drwxr-xr-x    7 marcoautili  staff   224 Sep 10  2018 Applications
 0 drwxr-xr-x@   5 marcoautili  staff   160 May  2 15:35 Applications (Parallels)
 0 drwx------@  58 marcoautili  staff  1856 Oct  1 17:11 Desktop
 0 drwx------+  11 marcoautili  staff   352 Dec 12  2018 Documents
 0 drwx------+ 164 marcoautili  staff  5248 Oct  1 16:57 Downloads
 0 drwx------@  18 marcoautili  staff   576 Jul 17 15:42 Dropbox
 0 drwx------@  78 marcoautili  staff  2496 Apr  4 14:12 Library
 0 drwx------@   7 marcoautili  staff   224 Sep 10  2018 Movies
 0 drwx------@   6 marcoautili  staff   192 Sep 10  2018 Music
 0 drwx------@   5 marcoautili  staff   160 Jan 20  2016 OneDrive - Universita' degli
Studi dell'Aquila
 0 drwx------    2 marcoautili  staff    64 May  2 15:24 Parallels
 0 drwx------+  10 marcoautili  staff   320 Sep 10  2018 Pictures
 0 drwxr-xr-x+   6 marcoautili  staff   192 May 23  2015 Public
 0 drwxr-xr-x@   3 marcoautili  staff    96 Sep 10  2018 Pydio
 0 drwxr-xr-x+   3 marcoautili  staff    96 Sep 10  2018 Sites
 0 drwxr-xr-x    7 marcoautili  staff   224 Sep 24  2017 SoapUI-Tutorials
 0 drwxr-xr-x    3 marcoautili  staff    96 Apr 19  2016 TEMP
 0 drwxr-xr-x    3 marcoautili  staff    96 Oct 23  2018 Wondershare
 8 -rw-r--r--    1 marcoautili  staff   453 Oct  8  2017 default-soapui-workspace.xml
 0 drwxr-xr-x    3 marcoautili  staff    96 Sep 27  2017 eclipse-workspace
 8 -rw-r--r--    1 marcoautili  staff   129 Sep 29  2016 pippo.txt
 8 -rw-r--r--    1 marcoautili  staff  2686 Oct  8  2017 soapui-settings.xml
iMac-di-User:~ marcoautili$

iMac-di-User:SoapUI-Tutorials marcoautili$ ls -ls
total 320
 16 -rw-r--r--  1 marcoautili  staff    7268 Nov 30  2016 Sample-Authentication-Project
 24 -rw-r--r--  1 marcoautili  staff    9456 Nov 30  2016 Sample-REST-Project-soapui-pr
280 -rw-r--r--  1 marcoautili  staff  142945 Nov 30  2016 Sample-SOAP-Project-soapui-pr
  0 drwxr-xr-x  4 marcoautili  staff     128 Sep 24  2017 WSDL-WADL
  0 drwxr-xr-x  8 marcoautili  staff     256 Sep 24  2017 popular-apis
iMac-di-User:SoapUI-Tutorials marcoautili$
iMac-di-User:SoapUI-Tutorials marcoautili$
```

Similarities and differences

MANPAGES

Index About Manpages FAQ Service Information

debian / bullseye / coreutils / ls(1)

```
-s, --size
          print the allocated size of each file, in blocks
-k, --kibibytes
          default to 1024-byte blocks for disk usage; used only with -s and
          per directory totals
```

archlinux™ ls Go

-s, --size

print the allocated size of each file, in blocks

-k, --kibibytes

default to 1024-byte blocks for file system usage; used only with **-s** and per directory totals

Desktop — less ◂ man ls — 84×38

From my macOS

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

NAME
     ls -- list directory contents

          -s      Display the number of file system blocks actually used by each
                  file, in units of 512 bytes, where partial units are rounded
                  up to the next integer value.  If the output is to a terminal,
                  a total sum for all the file sizes is output on a line before
                  the listing.  The environment variable BLOCKSIZE overrides the
                  unit size of 512 bytes.
```

Total (more on …)

- Starting from the early 2015, the MacBook comes with drives that use native advanced format 4096 bytes per sector drives
- Prior Macs use 512 bytes emulation even though the physical sectors are 4K

- Also, the Master File Table (MFT – more on file table on PART 4) on pre-2015 Mac drives use file record segments in 1024-byte units, whereas the 2015 Mac's drives use file record segments in 4096-byte segments

- For several years, hard drive manufacturers have been transitioning drives from 512 bytes sector size to 4096 bytes sector size. Hard drives have had physical sector sizes of 4096 bytes for many years, but the controllers in these drives present logical sector size as 512 bytes in emulation (512e) to support older computer hardware and operating systems

- After the 2015 change, Mac drive controllers present native 4096 bytes (advanced format) logical sectors to the operating system rather than 512 bytes

Total (more on …)

- Pre-2015 Macs generally show the Device Block Size (or logical sector size) as 512 bytes and 2015 Macs as 4096 bytes.
  - pre-2015_Mac$ diskutil info / | grep "Block Size"

    Device Block Size: 512 Bytes

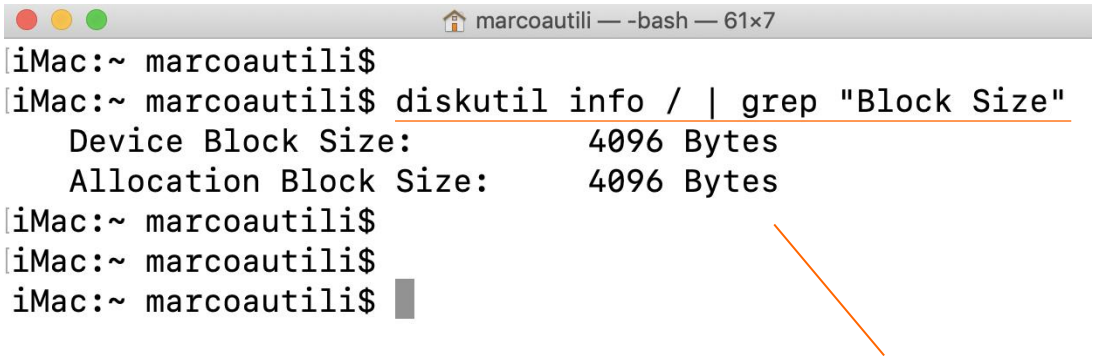
    Allocation Block Size: 4096 Bytes

  - 2015_Mac$ diskutil info / | grep "Block Size"

    Device Block Size: 4096 Bytes

    Allocation Block Size: 4096 Bytes

- The Device Block Size is the raw block size used by the hard drive controller hardware and cannot be changed
- The Allocation Block Size is used by the file system and is set when formatting and partitioning drives



Screenshot of my Mac after the transition to 4096 bytes (October 2019)

Total (more on …)

Screenshot of my Mac prior the transition to 4096 bytes in 2015

Screenshot of my Mac after the transition to 4096 bytes (October 2019)

Macintosh HD — -bash — 76×24

```
iMac-2:/ marcoautili$ pwd
/
iMac-2:/ marcoautili$ ls -l
total 45
drwxrwxr-x+ 76 root  admin  2584 Sep
drwxr-xr-x+ 67 root  wheel  2278 Dec
drwxr-xr-x@  2 root  wheel    68 Aug
drwxr-xr-x@  4 root  wheel   136 Sep
drwxr-xr-x   7 root  admin   238 Dec
drwxrwxrwt@  4 root  admin   136 Sep
drwxr-xr-x@ 39 root  wheel  1326 Aug
drwxrwxr-t@  2 root  admin    68 Aug
dr-xr-xr-x   3 root  wheel  4173 Sep
lrwxr-xr-x@  1 root  wheel    11 Oct
dr-xr-xr-x   2 root  wheel     1 Sep
-rw-r--r--@  1 root  wheel   313 Aug
dr-xr-xr-x   2 root  wheel     1 Sep
drwxr-xr-x   3 root  wheel   102 Dec
drwxr-xr-x@  6 root  wheel   204 Oct
drwxr-xr-x@ 59 root  wheel  2006 Aug
lrwxr-xr-x@  1 root  wheel    11 Oct
drwxr-xr-x@ 12 root  wheel   408 Dec
lrwxr-xr-x@  1 root  wheel    11 Oct
iMac-2:/ marcoautili$
```

marcoautili — -bash — 85×26

```
iMac-di-User:~ marcoautili$ ls -ls
total 24
0 drwxr-xr-x    7 marcoautili  staff   224 Sep 10  2018 Applications
0 drwxr-xr-x@   5 marcoautili  staff   160 May  2 15:35 Applications (Parallels)
0 drwx------@  58 marcoautili  staff  1856 Oct  1 17:11 Desktop
0 drwx------+  11 marcoautili  staff   352 Dec 12  2018 Documents
0 drwx------+ 164 marcoautili  staff  5248 Oct  1 16:57 Downloads
0 drwx------@  18 marcoautili  staff   576 Jul 17 15:42 Dropbox
0 drwx------@  78 marcoautili  staff  2496 Apr  4 14:12 Library
0 drwx------@   7 marcoautili  staff   224 Sep 10  2018 Movies
0 drwx------@   6 marcoautili  staff   192 Sep 10  2018 Music
0 drwx------@   5 marcoautili  staff   160 Jan 20  2016 OneDrive - Universita' degli
Studi dell'Aquila
0 drwx------    2 marcoautili  staff    64 May  2 15:24 Parallels
0 drwx------+  10 marcoautili  staff   320 Sep 10  2018 Pictures
0 drwxr-xr-x+   6 marcoautili  staff   192 May 23  2015 Public
0 drwxr-xr-x@   3 marcoautili  staff    96 Sep 10  2018 Pydio
0 drwxr-xr-x+   3 marcoautili  staff    96 Sep 10  2018 Sites
0 drwxr-xr-x    7 marcoautili  staff   224 Sep 24  2017 SoapUI-Tutorials
0 drwxr-xr-x    3 marcoautili  staff    96 Apr 19  2016 TEMP
0 drwxr-xr-x    3 marcoautili  staff    96 Oct 23  2018 Wondershare
8 -rw-r--r--    1 marcoautili  staff   453 Oct  8  2017 default-soapui-workspace.xml
0 drwxr-xr-x    3 marcoautili  staff    96 Sep 27  2017 eclipse-workspace
8 -rw-r--r--    1 marcoautili  staff   129 Sep 29  2016 pippo.txt
8 -rw-r--r--    1 marcoautili  staff  2686 Oct  8  2017 soapui-settings.xml
iMac-di-User:~ marcoautili$
```

Now it is clear why "45" is not a multiple of 8…

Extended attributes and ACL

- Extended attributes
- Access Control List

Macintosh HD — -bash — 76×24

```
iMac-2:/ marcoautili$ pwd
/
iMac-2:/ marcoautili$ ls -l
total 45
drwxrwxr-x+ 76 root  admin  2584 Sep  5 12:58 Applications
drwxr-xr-x+ 67 root  wheel  2278 Dec 18  2015 Library
drwxr-xr-x@  2 root  wheel    68 Aug 24  2015 Network
drwxr-xr-x@  4 root  wheel   136 Sep  5 13:01 System
drwxr-xr-x   7 root  admin   238 Dec 17  2015 Users
drwxrwxrwt@  4 root  admin   136 Sep  8 16:18 Volumes
drwxr-xr-x@ 39 root  wheel  1326 Aug 25 02:03 bin
```

Extended attributes and ACLs.
More on that later …

marcoautili — less ◂ man ls — 115×37

The Long Format

If the **-l** option is given, the following information is displayed for each file: file mode, number of links, owner name, group name, number of bytes in the file, abbreviated month, day-of-month file was last modified, hour file last modified, minute file last modified, and the pathname. In addition, for each directory whose contents are displayed, the total number of 512-byte blocks used by the files in the directory is displayed on a line by itself, immediately before the information for the files in the directory. If the file or directory has extended attributes, the permissions field printed by the **-l** option is followed by a '@' character. Otherwise, if the file or directory has extended security information (such as an access control list), the permissions field printed by the **-l** option is followed by a '+' character. If the **-%** option is given, a '%' character follows the permissions field for dataless files and directories, possibly replacing the '@' or '+' character.

https://en.wikipedia.org/wiki/Access-control_list

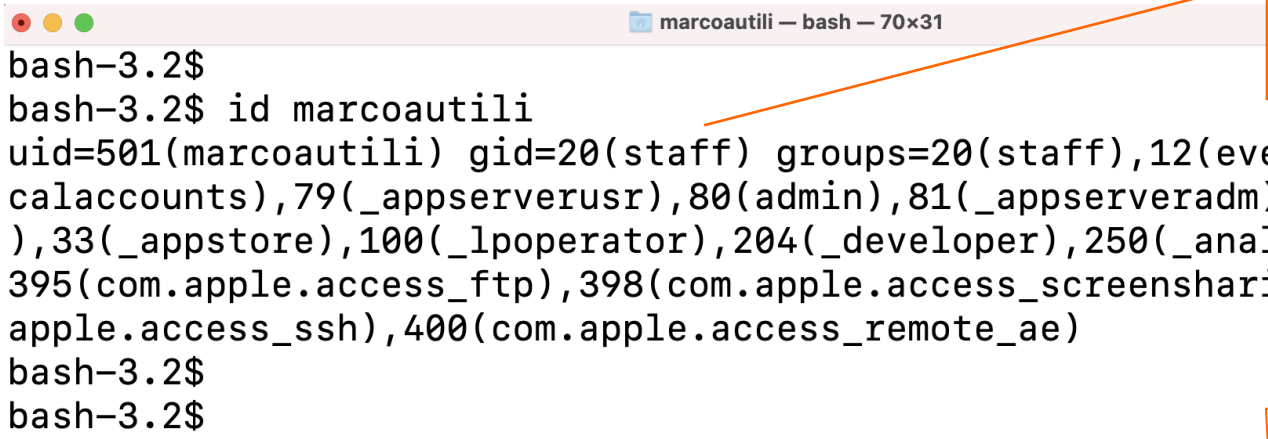
Users and groups

The id utility displays the user and group names and numeric IDs, of the calling process

- id -u display the User ID (UID)
- id -un display the username
- id -g display the primary (a.k.a. effective) Group ID (GID)
- id -gn display the primary Group Name
- id <login name or user ID> display user and group IDs of that user
- id -G display the IDs of all the groups
- id -Gn display the names of all the groups

The whoami utility has been obsoleted by the id utility, and is equivalent to id -un

On macOS, by default, your primary group is staff



more on that in Part 4

More on the Directory Service

dscacheutil
stands for Directory Service Cache Utility

Performs various operations against the Directory Service cache including gathering statistics, initiating lookups, inspection, cache flush, etc.

This tool replaces most of the functionality of the lookupd tool previously available

The Directory Service manages information about users and resources such printers and servers (Under macOS the Directory Service Architecture is called Open Directory)

Under UNIX-like or Linux systems you are probably used to modify files such /etc/passwd and /etc/group to create and modify users and groups

```
marcoautili — -bash — 62×32
iMac-2:~ marcoautili$
iMac-2:~ marcoautili$ dscacheutil -q group -a name staff
name: staff
password: *
gid: 20
users: root

iMac-2:~ marcoautili$
iMac-2:~ marcoautili$
iMac-2:~ marcoautili$ dscacheutil -q group -a name wheel
name: wheel
password: *
gid: 0
users: root

iMac-2:~ marcoautili$ dscl -q group -a name wheel
Cannot open remote host, error: DSOpenDirServiceErr
iMac-2:~ marcoautili$
iMac-2:~ marcoautili$ dscacheutil -q group -a name wheel
name: wheel
password: *
gid: 0
users: root

iMac-2:~ marcoautili$
iMac-2:~ marcoautili$ dscacheutil -q group -a name admin
name: admin
password: *
gid: 80
users: root marcoautili

iMac-2:~ marcoautili$
```

dscl -- Directory Service Command Line utility (see next slide)

This is not part of the course

More on the Directory Service

```
marcoautili — -bash — 45×25
iMac:~ marcoautili$
iMac:~ marcoautili$ dscl . list /groups
_amavisd
_analyticsd
_analyticsusers
_appleevents
_applepay
_appowner
_appserveradm
_appserverusr
_appstore
_ard
_assetcache
_astris
_atsserver
_calendar
_captiveagent
_ces
_clamav
_cmiodalassistants
_coreaudiod
```

List all groups

Directory Service Command Line utility

dscl is a general-purpose utility for operating on Directory Service directory nodes

Its commands allow one to create, read, and manage Directory Service data

```
marcoautili — dscl — 71×32
marcoautili@iMac ~ %
marcoautili@iMac ~ % dscl
Entering interactive mode... (type "help" for commands)
 >
 >
```

If invoked without any commands, dscl runs in an interactive mode, reading commands from standard input

```
marcoautili — -bash — 64×38
iMac-2:~ marcoautili$ dscacheutil -q user -a name root
name: root
password: *
uid: 0
gid: 0
dir: /var/root
shell: /bin/sh
gecos: System Administrator

iMac-2:~ marcoautili$ dscacheutil -q user
name: _amavisd
password: *
uid: 83
gid: 83
dir: /var/virusmails
shell: /usr/bin/false
gecos: AMaViS Daemon

name: _appleevents
password: *
uid: 55
gid: 55
dir: /var/empty
shell: /usr/bin/false
gecos: AppleEvents Daemon

name: _appowner
password: *
uid: 87
gid: 87
dir: /var/empty
shell: /usr/bin/false
gecos: Application Owner

name: _appserver
password: *
uid: 79
gid: 79
```

Lookup a user

Lookup all users

This is not part of the course

Homework Activities

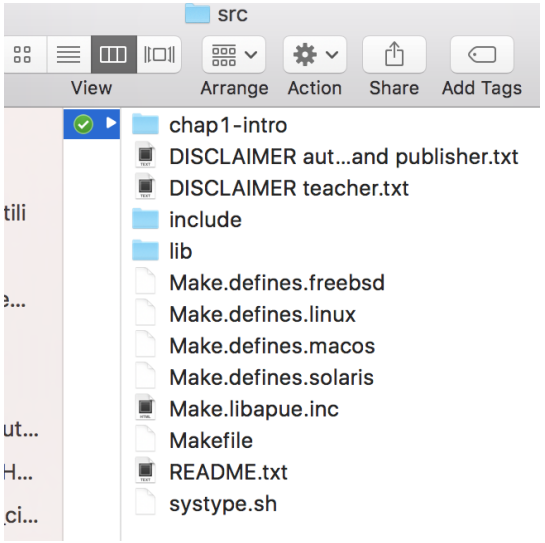
Use the commands cd and ls to navigate your system and explore your directories by using different methods and different relative and absolute paths

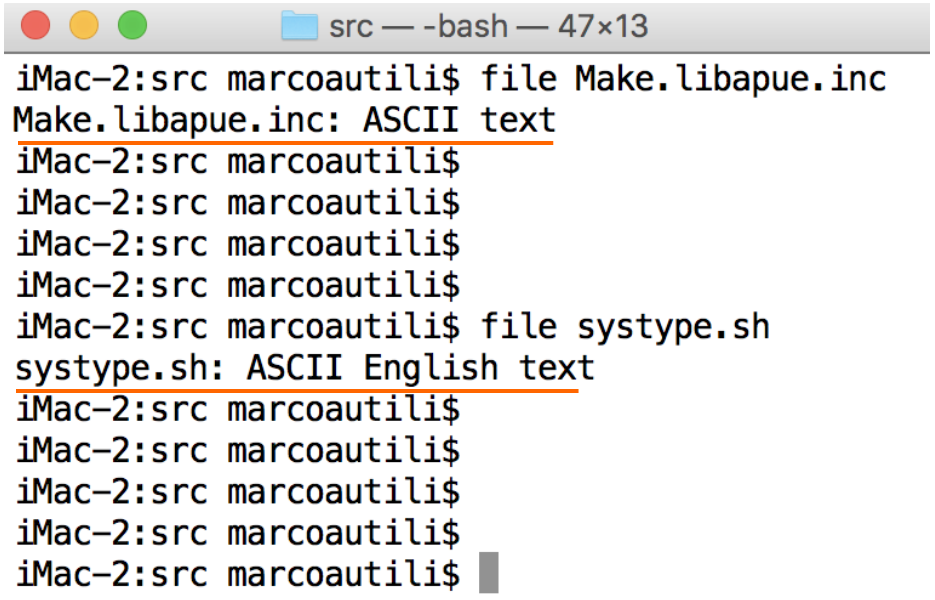
For example

- /etc - the place where config files for the system are stored
- /var/log - the place where log files for various system programs are stored (You may not have the right permissions…)
- /bin - the place where many programs are contained
- /usr/bin - system programs are also stored here

Everything is a File

- A directory is a file
- Mouse, keyboard and monitor are files
  - the keyboard is a read-only file!
  - the monitor is write-only file!
- Windows uses the extension (.txt, .png) of a file to understand what type of file it is
- Linux/Unix-like systems are said to be "extensionless" systems, i.e., the system looks inside the file to understand what type of file it is
  - The command file [path] is of help





Case Sensitive VS Case Insensitive

Historically,

- UNIX-like and Linux systems are Case Sensitive (e.g., pippo.txt is distinct from PIPPO.txt)
- Window systems is Case Insensitive
- Linux is Case Sensitive also with command line options
  - ls command
    - ls -S Sort files by size
    - ls -s (as already said) Display the number of file system blocks actually used by each file, in units of 512 bytes, where partial units are rounded up to the next integer value. If the output is to a terminal, a total sum for all the file sizes is output on a line before the listing. The environment variable BLOCKSIZE (when set) overrides the unit size of 512 bytes
- macOS systems (which are Unix-based) typically have case-insensitive file systems (i.e., pippo.txt is the same as PIPPO.txt) so on such systems commands external to the shell are in fact treated case-insensitively. But builtins, like cd, remain case-sensitive (more on builtin commands later)

Again, many similarities together with many differences!

Special chars in file and directory names

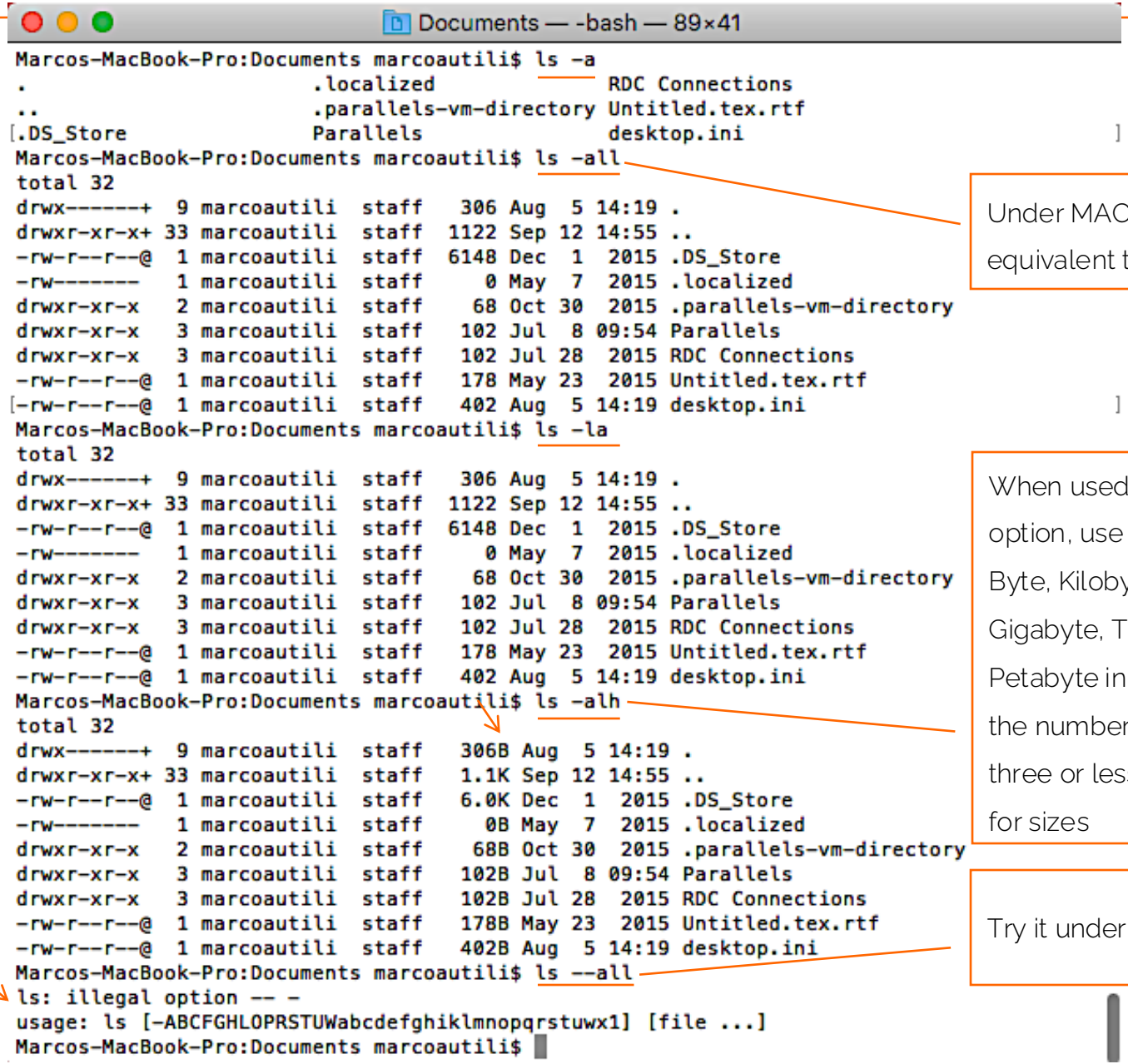
Spaces in file and directory names are valid… but be careful!

Two way to deal with

1. Quotes (single or double not a problem for now…)
   - cd 'Applications (Parallels)'/
   - cd "Applications (Parallels)"/
2. Escape character
   - cd Applications\ \(Parallels\)/
     - Note that three chars have been escaped with \, i.e., space, (and)

Remember
→| Tab Completion automatically escapes chars for you!

Long hand and short-hand version



Under MAC OSX, it is equivalent to ls –al or ls -la

When used with the -l option, use unit suffixes: Byte, Kilobyte, Megabyte, Gigabyte, Terabyte and Petabyte in order to reduce the number of digits to three or less using base 2 for sizes

Try it under Linux…

$PATH

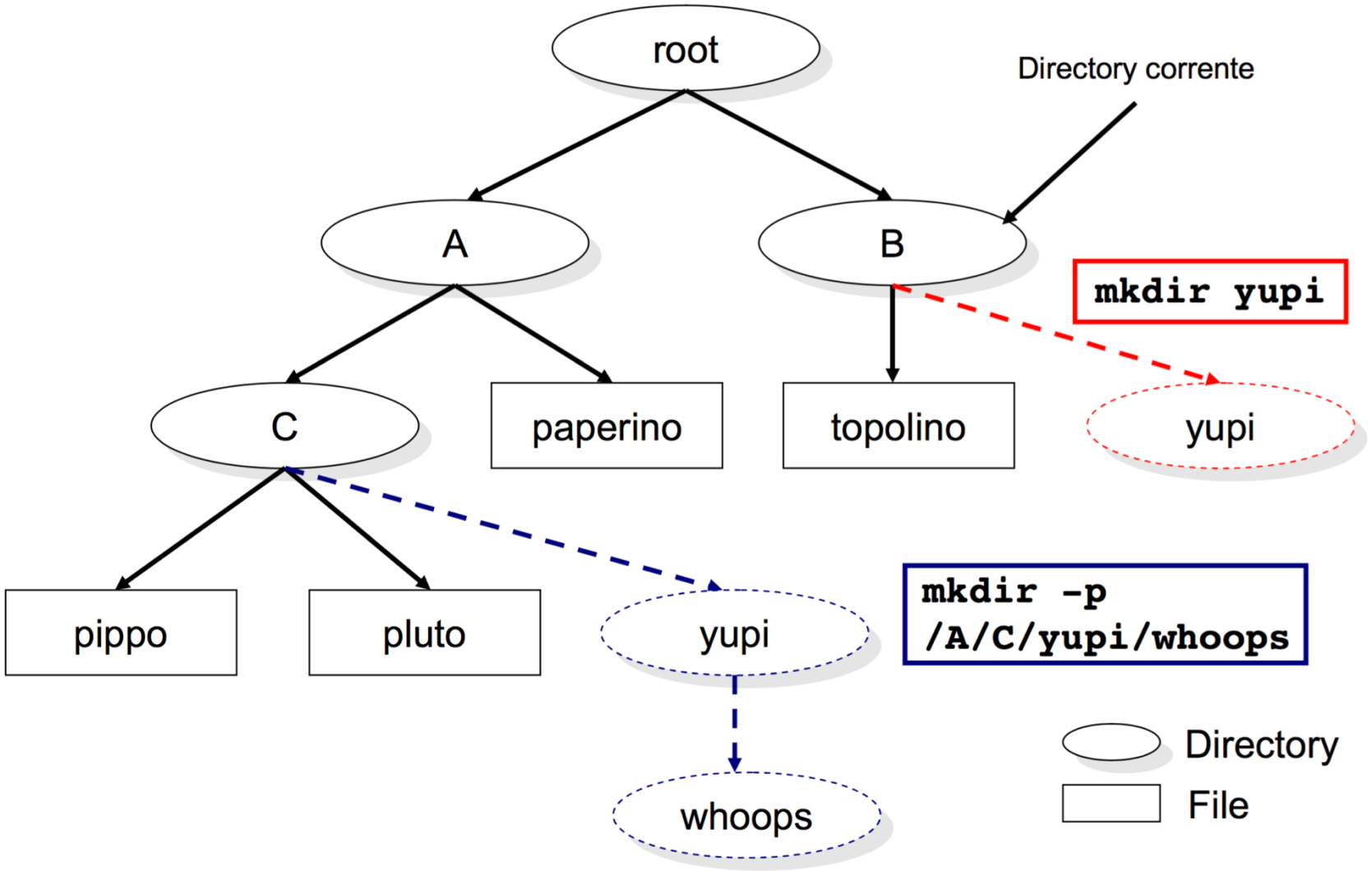
- When we type a command on the terminal, the system searches for the implementation of the command in a predefined set of directories
  - The predefined set of directories to be searched is specified by the variable $PATH

- The system will not look inside the current directory, that's why when we want to run our own script from within the current directory, we need to use "./"
  - ./myfirstscript.sh

More on that in PART III

```
bin — -bash — 74×47
[vincet:~ marcoautili$ echo $PATH
 /usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/texbin
[vincet:~ marcoautili$
[vincet:~ marcoautili$ cd /bin
[vincet:bin marcoautili$
[vincet:bin marcoautili$ ls -la
 total 5168
 drwxr-xr-x@ 39 root  wheel     1326 Aug  3 11:09 .
 drwxr-xr-x  34 root  wheel     1224 Sep 21 13:24 ..
 -rwxr-xr-x   1 root  wheel    22464 Dec  3  2015 [
 -r-xr-xr-x   1 root  wheel   628496 Dec  3  2015 bash
 -rwxr-xr-x   1 root  wheel    23520 Dec  3  2015 cat
 -rwxr-xr-x   1 root  wheel    33904 Jul  9 04:52 chmod
 -rwxr-xr-x   1 root  wheel    28832 Jul  9 04:53 cp
 -rwxr-xr-x   1 root  wheel   378624 Mar 12  2016 csh
 -rwxr-xr-x   1 root  wheel    28368 Dec  3  2015 date
 -rwxr-xr-x   1 root  wheel    31856 Jul  9 04:52 dd
 -rwxr-xr-x   1 root  wheel    27440 Jul  9 04:52 df
 -r-xr-xr-x   1 root  wheel    18144 Dec  3  2015 domainname
 -rwxr-xr-x   1 root  wheel    18032 Dec  3  2015 echo
 -rwxr-xr-x   1 root  wheel    53872 Dec  3  2015 ed
 -rwxr-xr-x   1 root  wheel    22992 Dec  3  2015 expr
 -rwxr-xr-x   1 root  wheel    18192 Dec  3  2015 hostname
 -rwxr-xr-x   1 root  wheel    18528 Dec  3  2015 kill
 -r-xr-xr-x   1 root  wheel  1394432 Mar 12  2016 ksh
 -rwxr-xr-x   1 root  wheel   124048 Jul  9 04:52 launchctl
 -rwxr-xr-x   1 root  wheel    18944 Jul  9 04:52 link
 -rwxr-xr-x   1 root  wheel    18944 Jul  9 04:52 ln
 -rwxr-xr-x   1 root  wheel    38512 Jul  9 04:52 ls
 -rwxr-xr-x   1 root  wheel    18496 Jul  9 04:53 mkdir
 -rwxr-xr-x   1 root  wheel    24144 Jul  9 04:52 mv
 -rwxr-xr-x   1 root  wheel   110800 Jul  9 04:52 pax
 -rwsr-xr-x   1 root  wheel    51008 Dec  3  2015 ps
 -rwxr-xr-x   1 root  wheel    18176 Dec  3  2015 pwd
 -r-xr-xr-x   1 root  wheel    29520 Dec  3  2015 rcp
 -rwxr-xr-x   1 root  wheel    23744 Jul  9 04:52 rm
 -rwxr-xr-x   1 root  wheel    18064 Jul  9 04:52 rmdir
 -r-xr-xr-x   1 root  wheel   632672 Dec  3  2015 sh
 -rwxr-xr-x   1 root  wheel    17984 Dec  3  2015 sleep
 -rwxr-xr-x   1 root  wheel    32048 Dec  3  2015 stty
 -rwxr-xr-x   1 root  wheel    42320 Jul  9 04:52 sync
 -rwxr-xr-x   1 root  wheel   378624 Mar 12  2016 tcsh
 -rwxr-xr-x   1 root  wheel    22464 Dec  3  2015 test
 -rwxr-xr-x   1 root  wheel    23744 Jul  9 04:52 unlink
 -rwxr-xr-x   1 root  wheel    18080 Jul  9 04:52 wait4path
 -rwxr-xr-x   1 root  wheel   573600 Dec  3  2015 zsh
 vincet:bin marcoautili$
```

Manipulating files and directories



mkdir & rmdir

mkdir [options] <DirectoryPath>

- mkdir pluto OR mkdir ./pluto
- mkdir /Users/marcoautili/Documents/pluto
- mkdir ../pluto
- mkdir ~/pluto

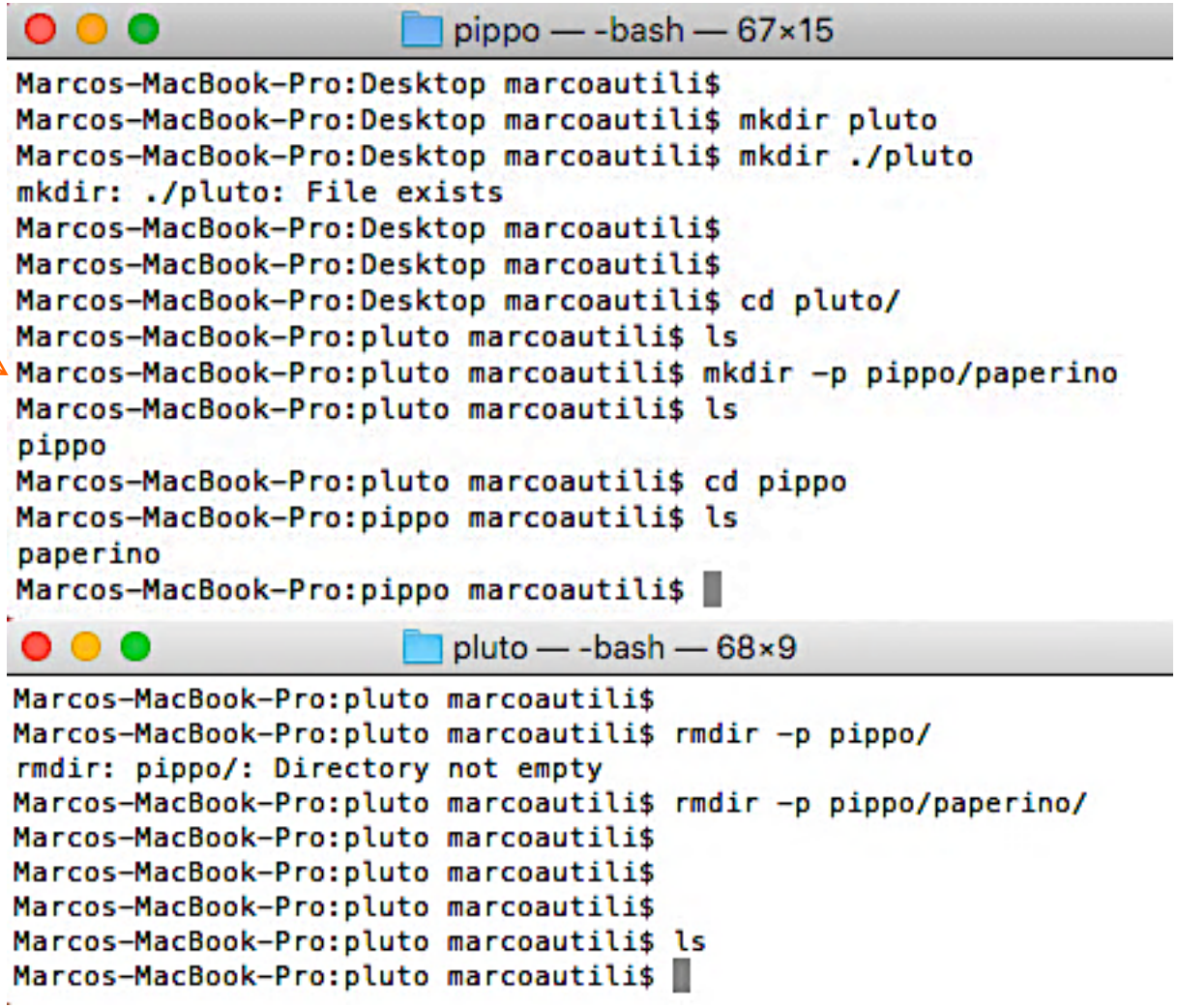
mkdir -p <DirectoryPath>

Create intermediate directories as required

rmdir [options] <DirectoryPath>

Remove a directory if it is empty

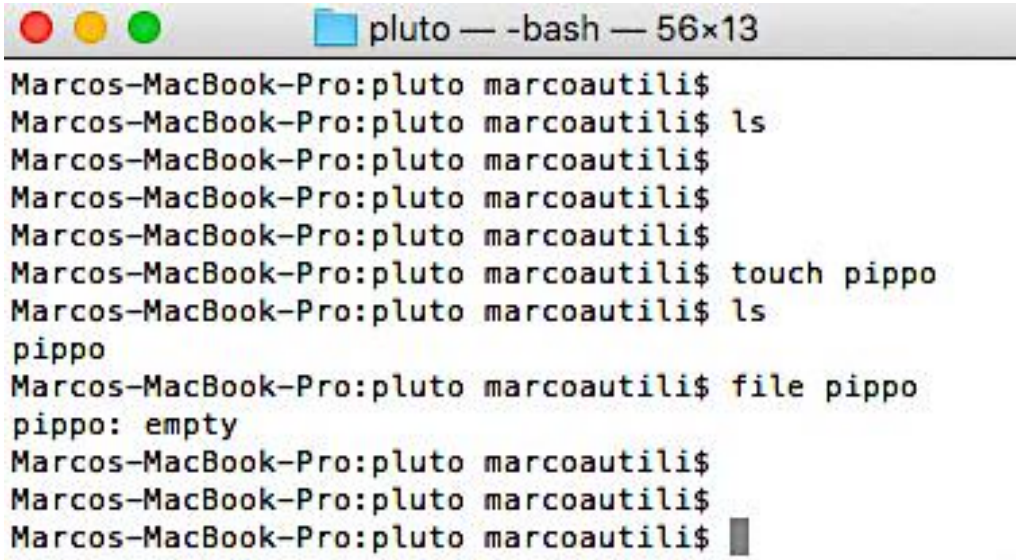
See also later slides



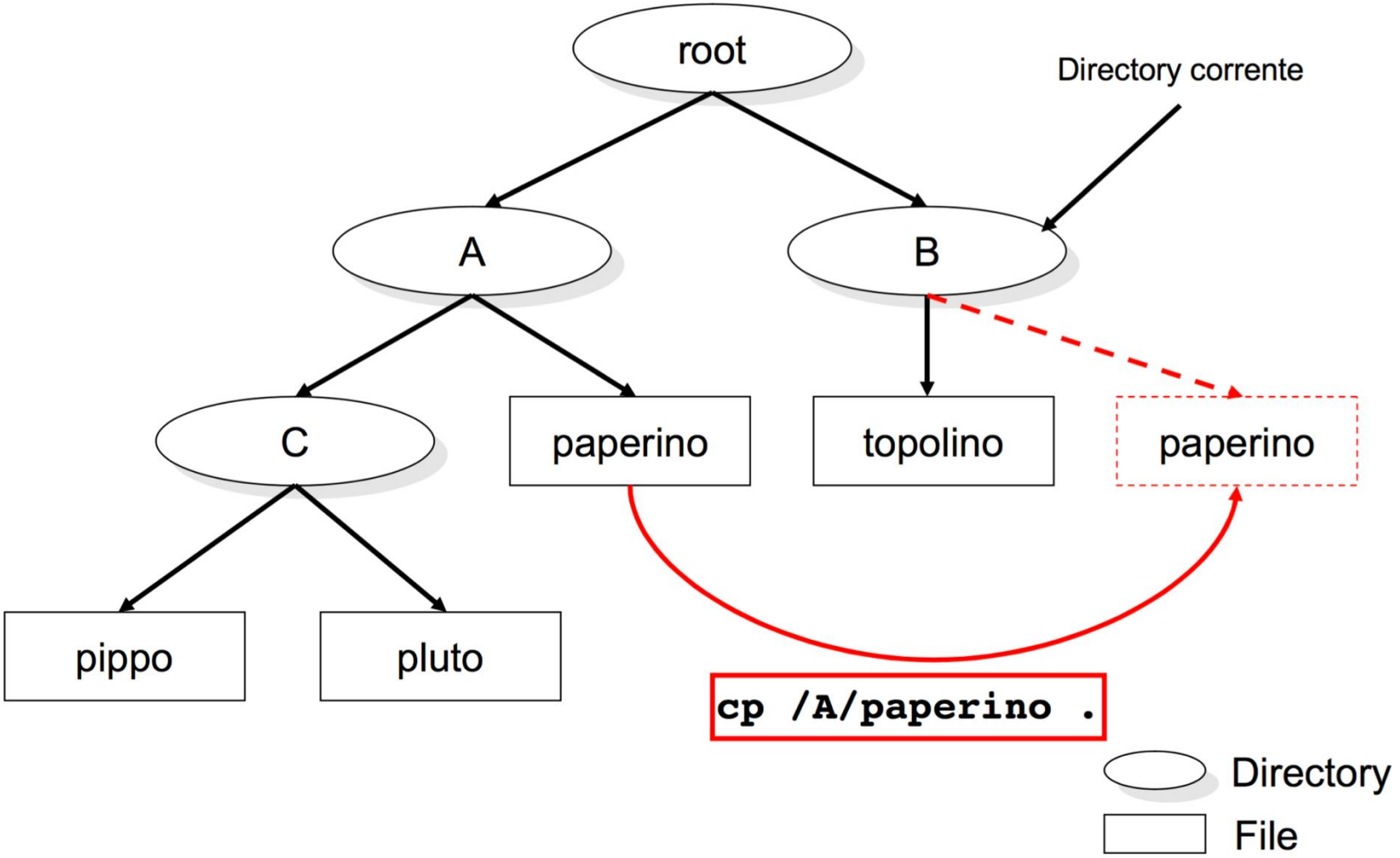
Creating an empty file

touch [options] <filename>

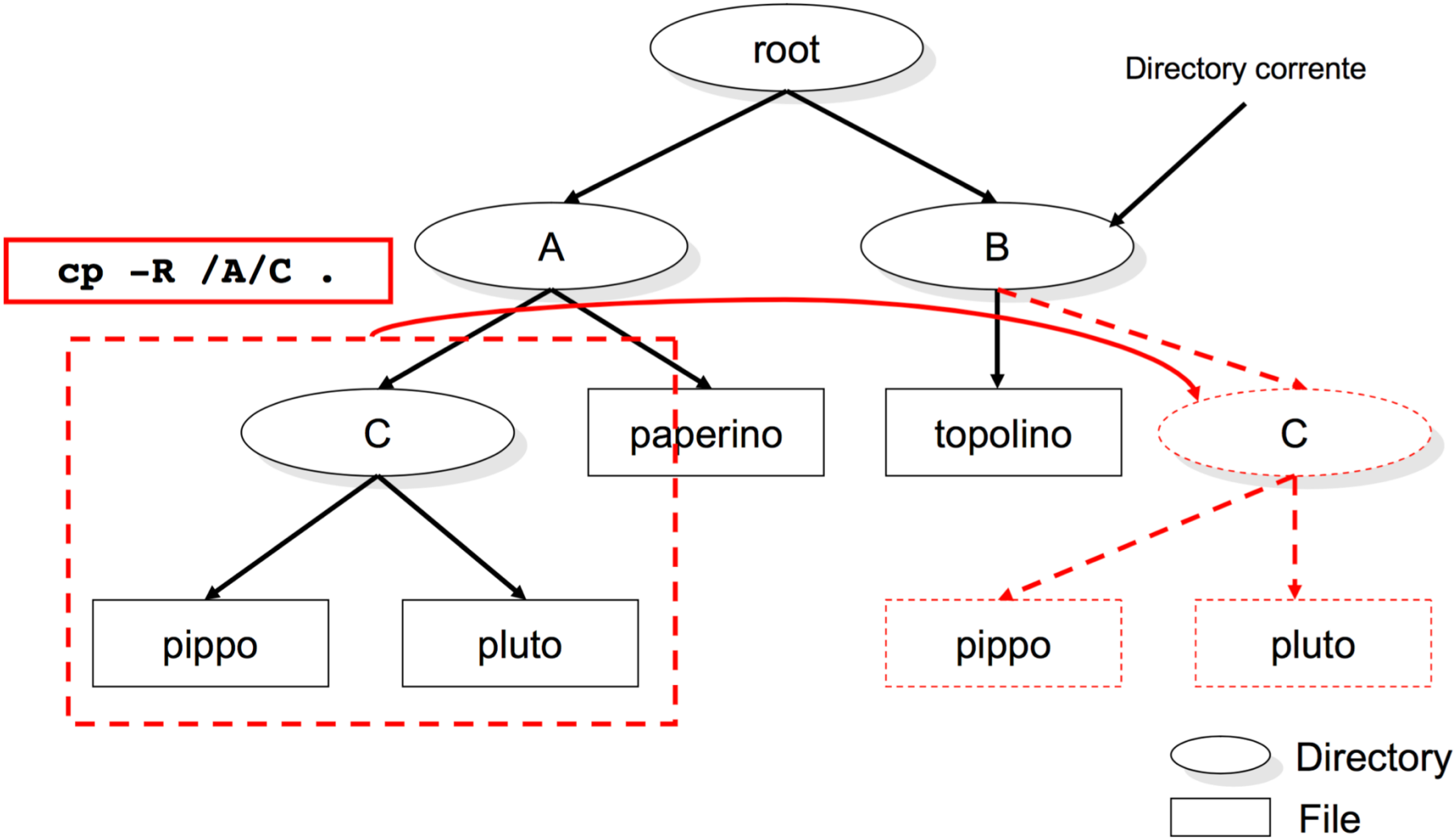
- change file access and modification times
  - It is not for creating files, but it creates an empty file if the specified file <filename> does not exist



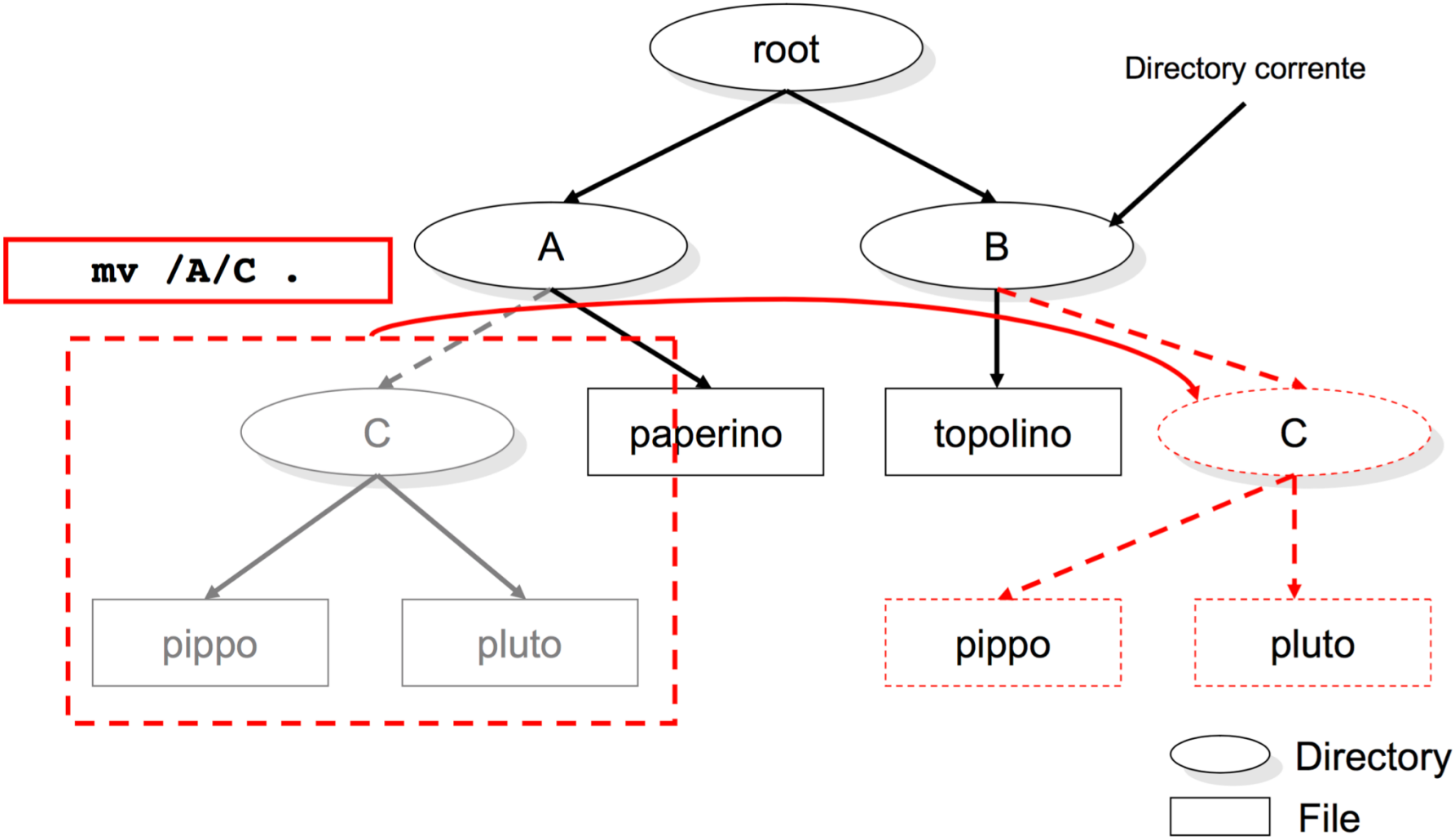
Copying


root
Directory corrente
A
B
C
paperino
topolino
paperino
pippo
pluto
cp /A/paperino .
Directory
File

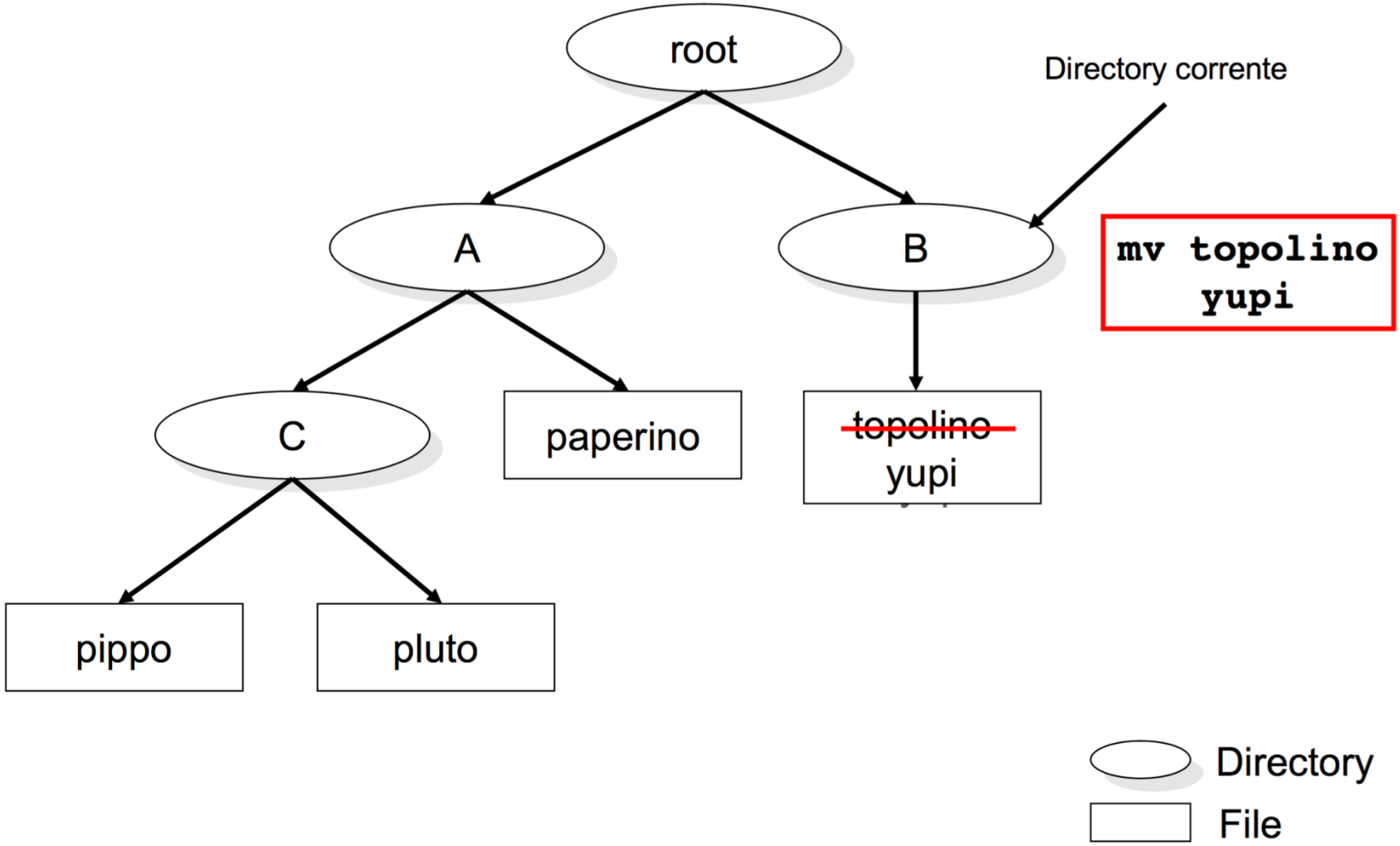
Copying


root
Directory corrente
A
B
cp -R /A/C .
C
paperino
topolino
C
pippo
pluto
pippo
pluto
Directory
File

Moving

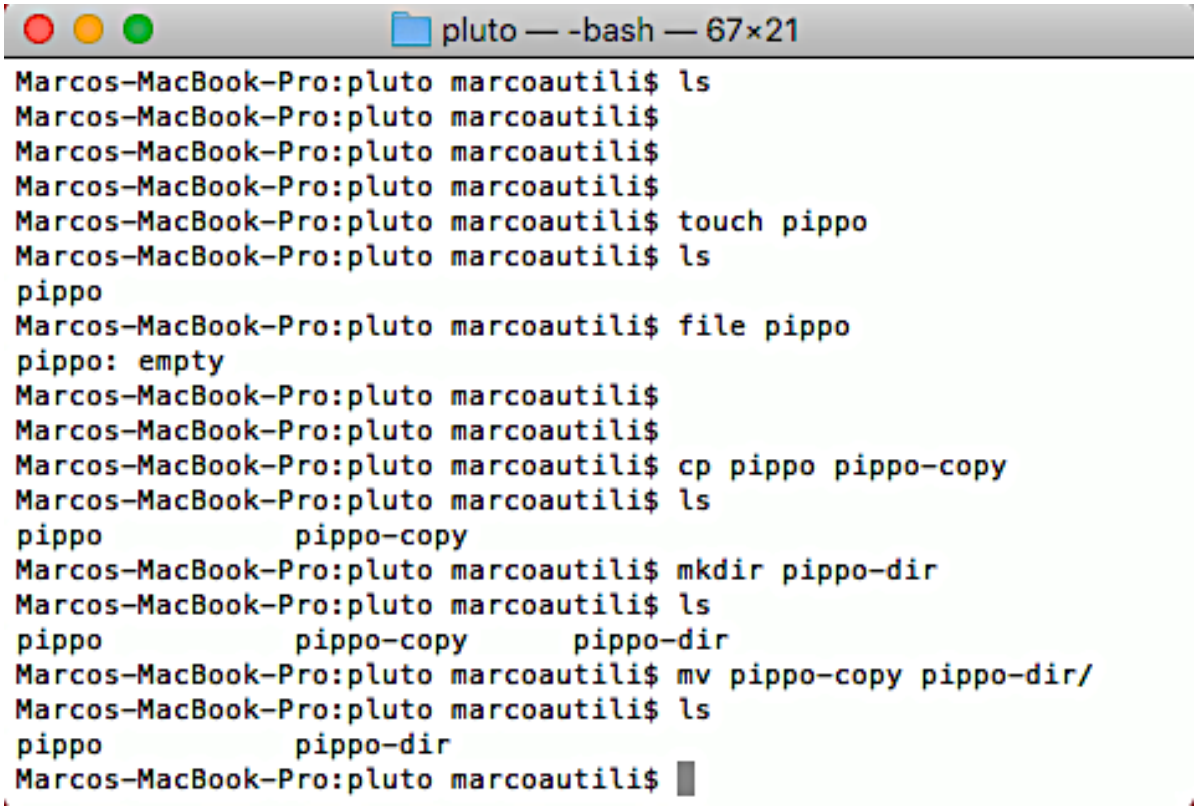

root
Directory corrente
A
B
mv /A/C .
C
paperino
topolino
C
pippo
pluto
pippo
pluto
Directory
File

Moving (when used for renaming)



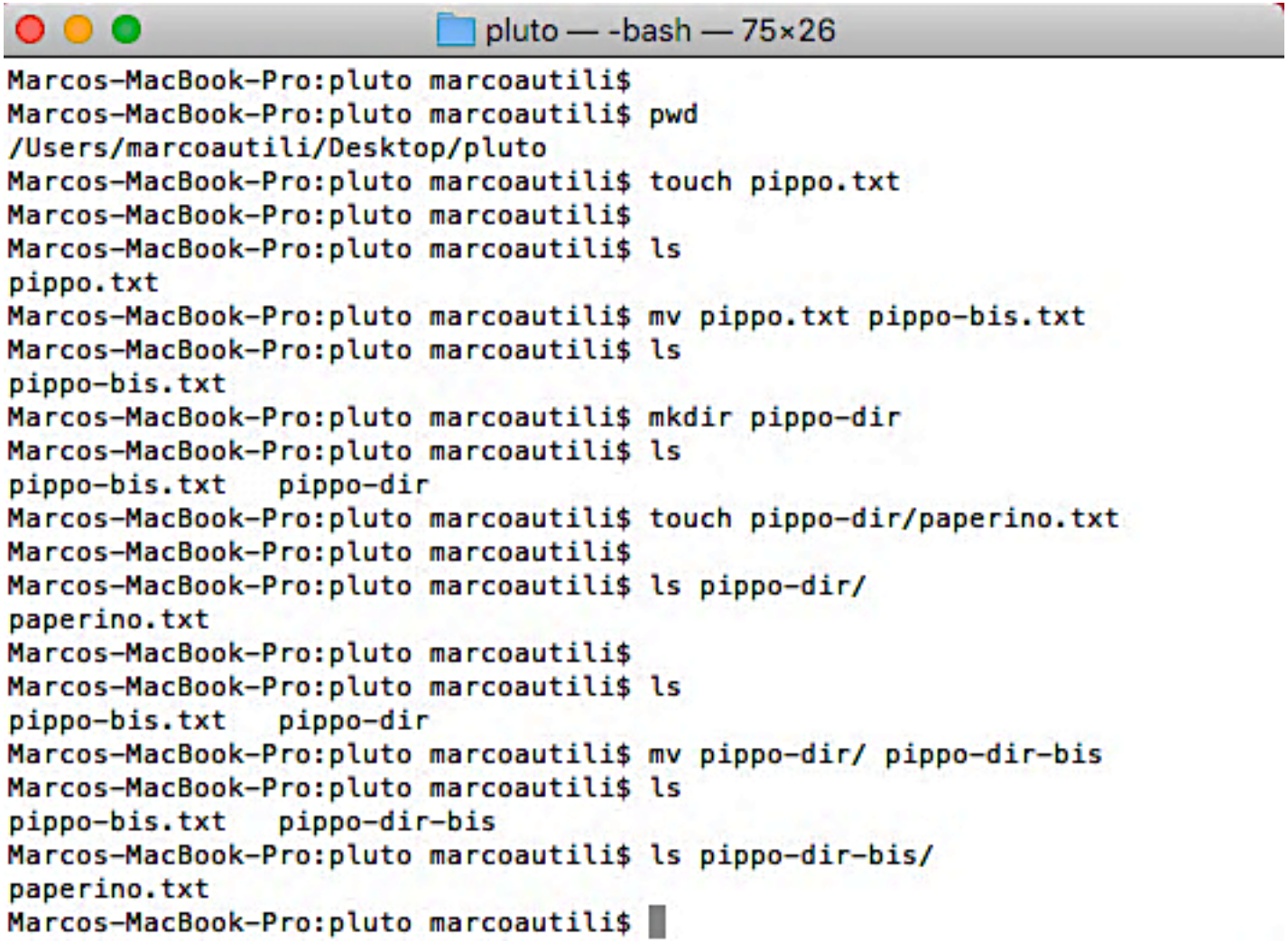
cp & mv

- cp [options] <sourcePath> <destinationPath>
  - many many options… see the dedicated man page
  - use the -r option (i.e., recursive) to copy directories
- mv [options] <source> <destination>
  - directories can be moved without the -r option



Renaming

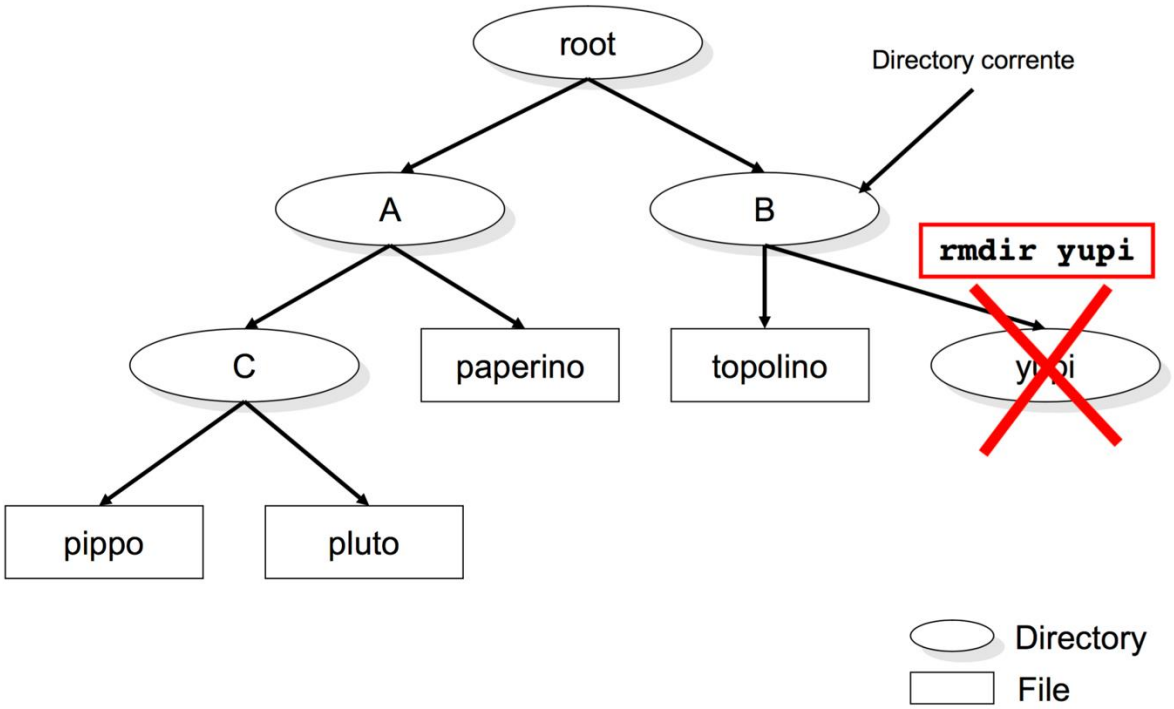
- mv [options] <source> <destination>
  - Note that it is not for renaming but it can be used for renaming both files and directories



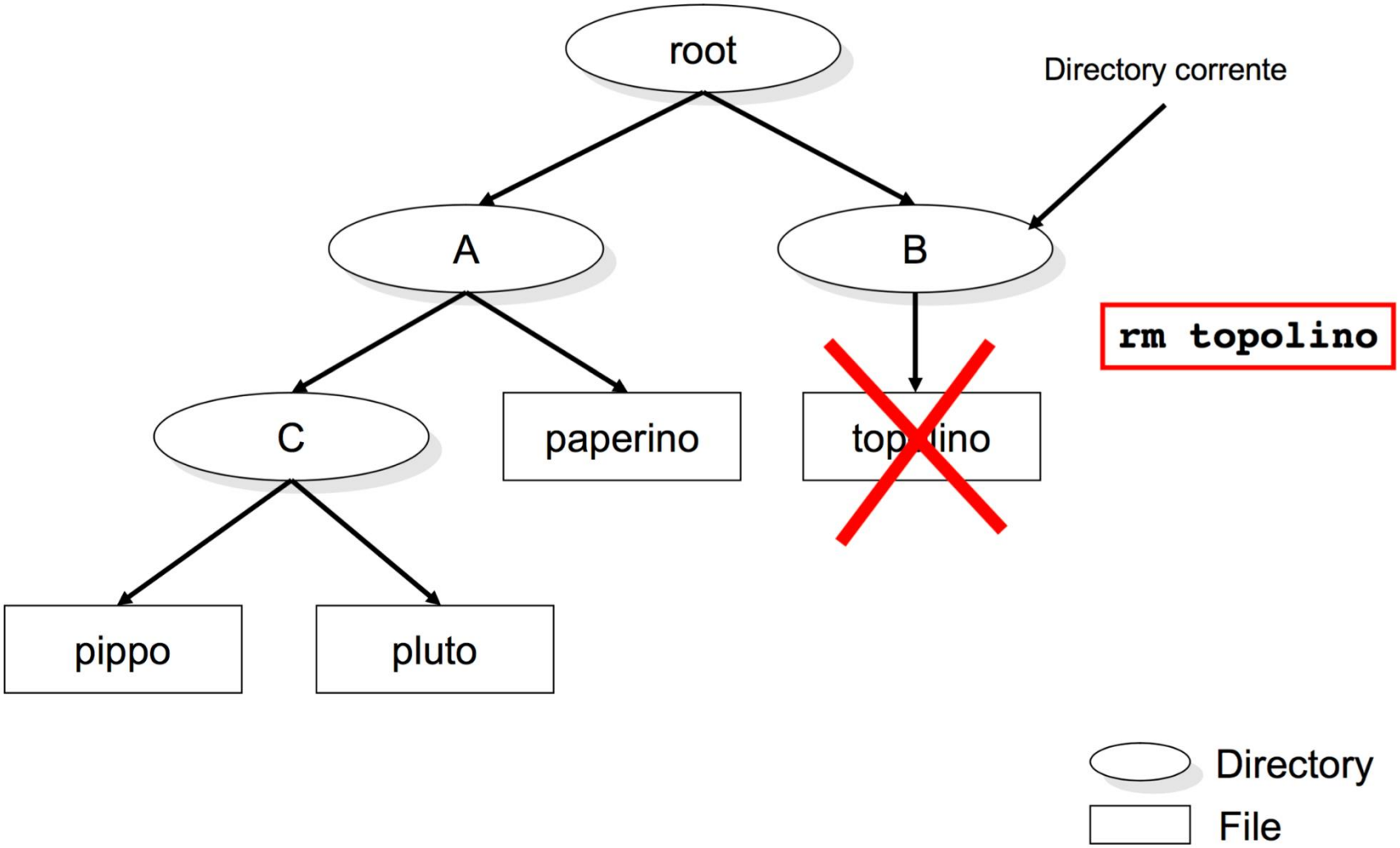
Removing empty directories

rmdir [-p] <pathname>

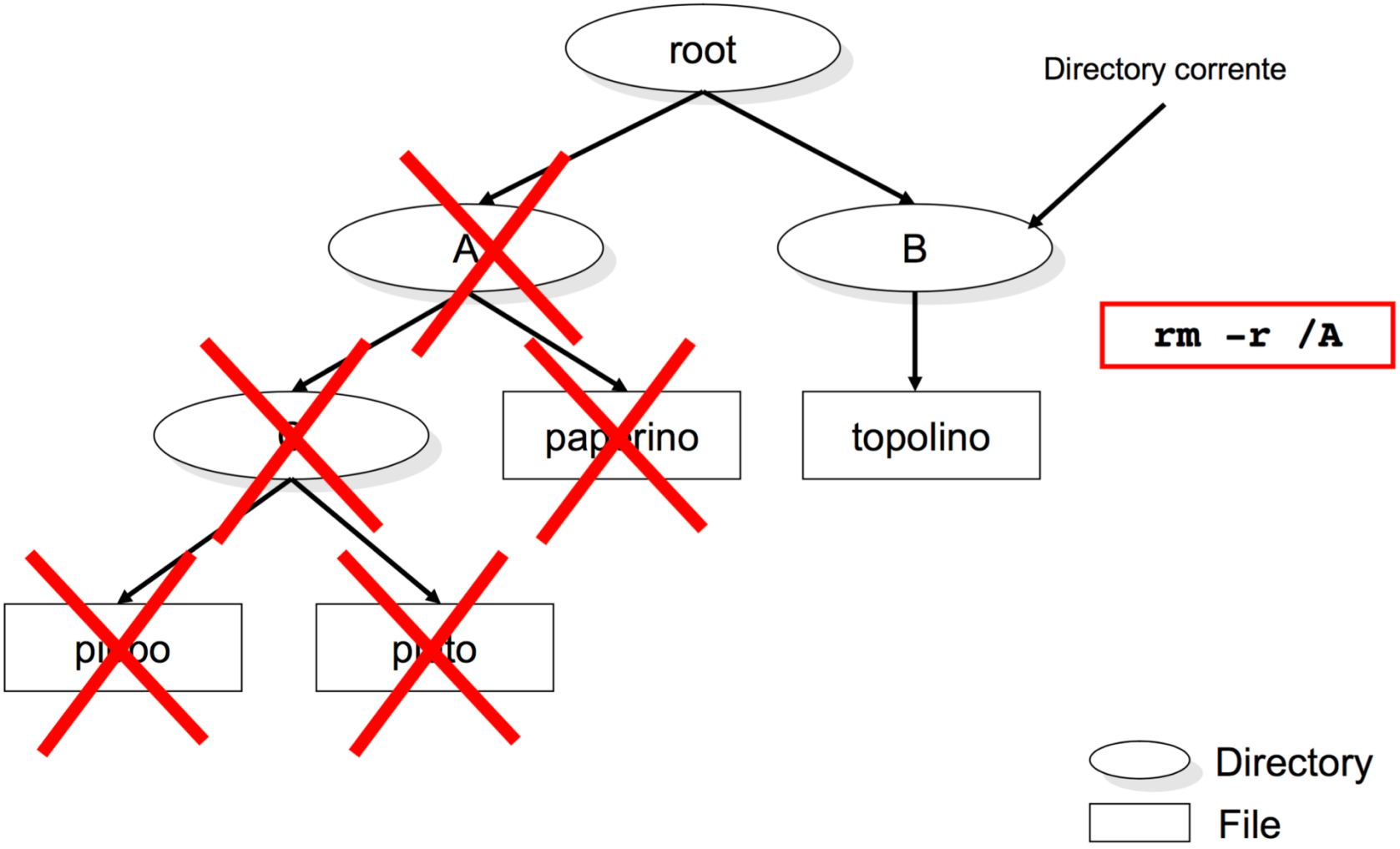
- remove empty directories
- The –p option tries to remove all the directories specified by the pathname,
  - e.g., rmdir -p a/c is equivalent to rmdir a/c a



Removing files



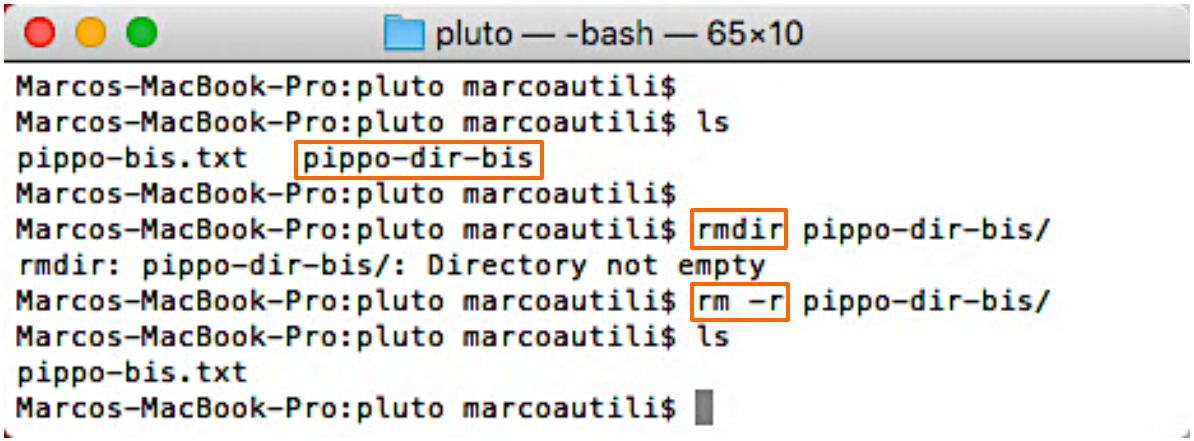
Removing not empty directories



rmdir versus rm

rm [options] <file>

- many many options… see the dedicated man page
- similar to cp for what concerns the -r option



Removing

- Be careful when using rm –r *

More on wildcards later

  - a good practice would be to use the option i (i.e., interactive) together with r
  - This option will prompt you before removing each file and directory and give you the option to

```
pluto — -bash — 65×11
Marcos-MacBook-Pro:pluto marcoautili$
Marcos-MacBook-Pro:pluto marcoautili$ ls
pippo-bis.txt   pippo-dir-bis
Marcos-MacBook-Pro:pluto marcoautili$ ls pippo-dir-bis/
paperino.txt
Marcos-MacBook-Pro:pluto marcoautili$ rm -ri pippo-dir-bis/
examine files in directory pippo-dir-bis/? y
remove pippo-dir-bis//paperino.txt? n
remove pippo-dir-bis/? y
rm: pippo-dir-bis/: Directory not empty
Marcos-MacBook-Pro:pluto marcoautili$
```

To mitigate the "destructive" nature of "rm", the system can be configured in such a way that "rm" is aliased to "rm -i"

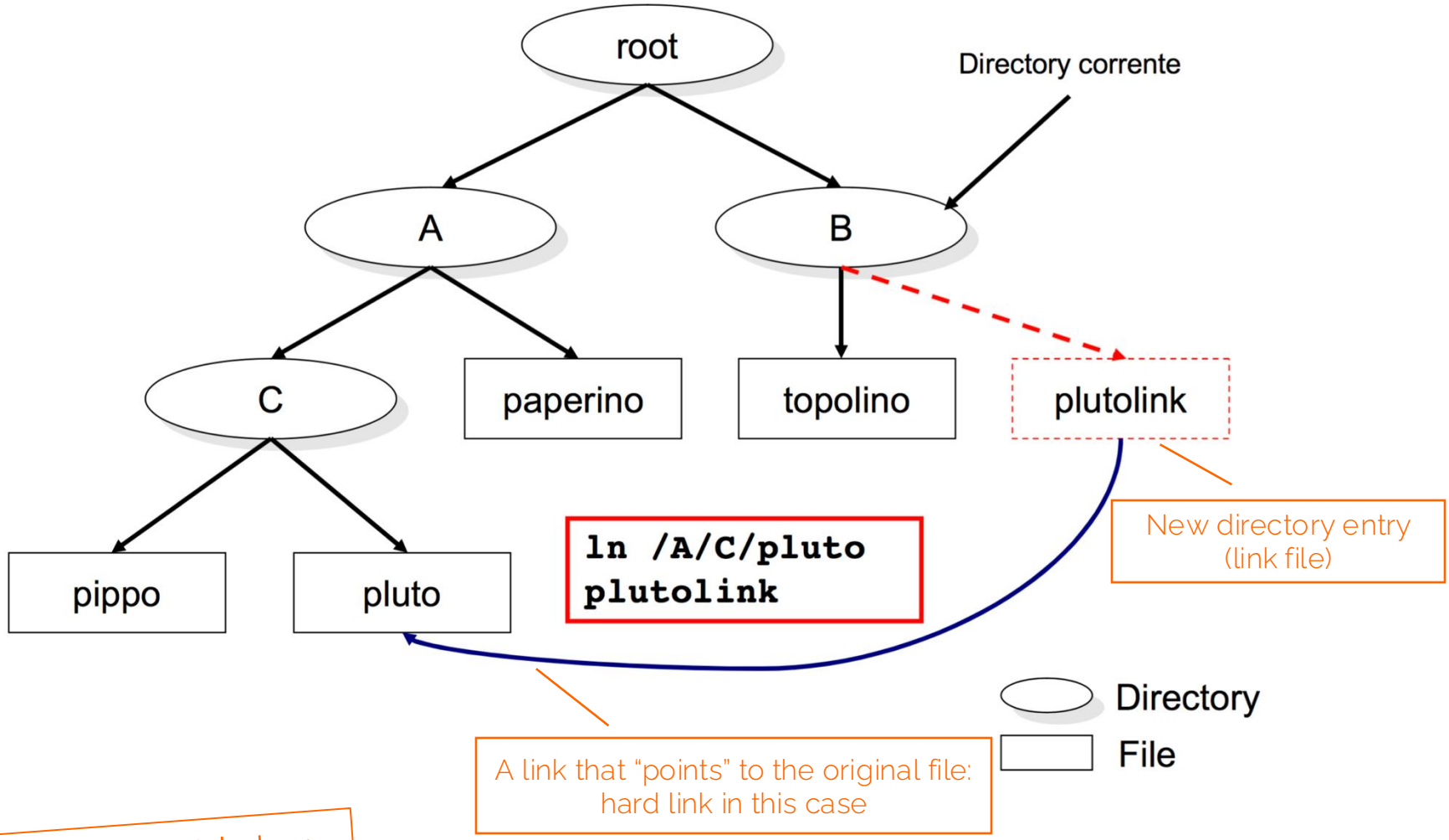
See next slide...

```
pluto — -bash — 61×19
remove pippo-dir-bis//paperino.txt? n
remove pippo-dir-bis/? y
rm: pippo-dir-bis/: Directory not empty
Marcos-MacBook-Pro:pluto marcoautili$
Marcos-MacBook-Pro:pluto marcoautili$
Marcos-MacBook-Pro:pluto marcoautili$
Marcos-MacBook-Pro:pluto marcoautili$ ls
pippo-bis.txt   pippo-dir-bis
Marcos-MacBook-Pro:pluto marcoautili$
Marcos-MacBook-Pro:pluto marcoautili$
Marcos-MacBook-Pro:pluto marcoautili$ rm -ri pippo-dir-bis/
examine files in directory pippo-dir-bis/? y
remove pippo-dir-bis//paperino.txt? y
remove pippo-dir-bis/? y
Marcos-MacBook-Pro:pluto marcoautili$
Marcos-MacBook-Pro:pluto marcoautili$
Marcos-MacBook-Pro:pluto marcoautili$ ls
pippo-bis.txt
Marcos-MacBook-Pro:pluto marcoautili$
```

Command aliasing

```
src — -bash — 108×31
desktop-jisqlks:src marcoautili$ ls
DISCLAIMER authors and publisher.txt    Make.defines.solaris                include
DISCLAIMER teacher.txt                  Make.libapue.inc                    lib
Make.defines.freebsd                    Makefile                            systype.sh
Make.defines.linux                      README.txt
Make.defines.macos                      chap1-intro
desktop-jisqlks:src marcoautili$
desktop-jisqlks:src marcoautili$
desktop-jisqlks:src marcoautili$ touch pippo-file-to-be-deleted.txt
desktop-jisqlks:src marcoautili$
desktop-jisqlks:src marcoautili$ ls
DISCLAIMER authors and publisher.txt    Make.defines.solaris                include
DISCLAIMER teacher.txt                  Make.libapue.inc                    lib
Make.defines.freebsd                    Makefile                            pippo-file-to-be-deleted.txt
Make.defines.linux                      README.txt                          systype.sh
Make.defines.macos                      chap1-intro
desktop-jisqlks:src marcoautili$
desktop-jisqlks:src marcoautili$ alias rm='rm -i'
desktop-jisqlks:src marcoautili$
desktop-jisqlks:src marcoautili$ rm pippo-file-to-be-deleted.txt
remove pippo-file-to-be-deleted.txt? yes
desktop-jisqlks:src marcoautili$
desktop-jisqlks:src marcoautili$ ls
DISCLAIMER authors and publisher.txt    Make.defines.solaris                include
DISCLAIMER teacher.txt                  Make.libapue.inc                    lib
Make.defines.freebsd                    Makefile                            systype.sh
Make.defines.linux                      README.txt
Make.defines.macos                      chap1-intro
desktop-jisqlks:src marcoautili$
desktop-jisqlks:src marcoautili$ unalias rm
desktop-jisqlks:src marcoautili$
```

(Hard) Link



More on that later…

Cat and Less commands

- cat <file> (stands for concatenate)
  - Its main purpose if to concatenate files together
  - It can be conveniently used for viewing files
  - However, it is appropriate to view small files

- less <file>
  - more appropriate for lager files
  - arrow keys can be used to move up and down within a file
  - SpaceBar to move forward a whole page
  - b to move back a whole page
  - q quit the file

- see also more <file>

Homework Activities

- Use the commands we have learnt so far to create, delete, copy, move, rename files and directories

- Navigate the file system and check the created directories and files

- Be careful with commands without undo features
  - i.e., be careful when performing destructive actions… command aliasing can be of help!

Editing files

Editing a file is to modify the content of a file

- Text editor
  - Enter and modify text in a text file
- Word processor
  - Enter, modify and format text in a document
- Line editor
  - Edit file one line at a time
  - Unix examples: ex, ed and sed
- Full screen editor
  - Shows a whole screen of text at a time

The following slides on editors will not be part of the exam

Editor features

- enter text

- search and replace
- copy, cut and paste
- undo and redo

- importing and exporting text
- save and cancel

Text files

- As you already know, Unix file name does not require file extension
  - Unix file system does not consider the extension when treating files

- However, some extensions are commonly used
  - Program source code: .c .cc .cpp .f .f77 .f95
  - Compiled object code: .o .a .so .sa
  - Compressed files: .z .gz .zip
  - Archive files: .tar .tz
  - Web site source code: .html .shtml .php
  - Notably, executable files typically have no extension

Unix Text editors

- vi
- vim (vi improved)
- emacs
- pico
- nano

- GUI editors
  - emacs
  - mousepad
  - Xedit

History on wikipedia…

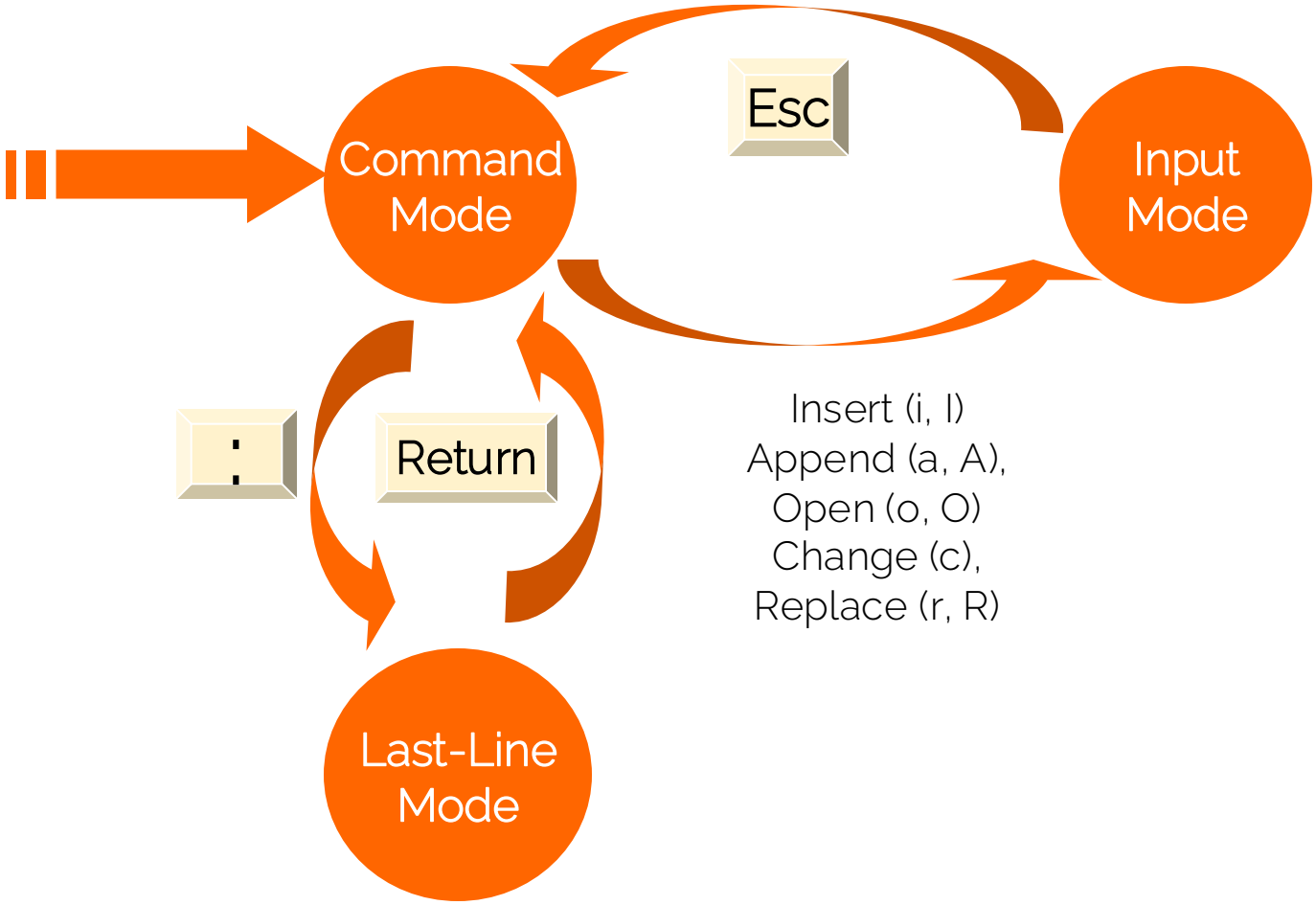
vi Text Editor

- A command line text editor
- A plain text editor
- Not a word processor
- Learning it might be "a bother"
- Useful and powerful
  - (more than Notepad on Window or Textedit on Mac)
- No GUI… forget the mouse

- vi has multiple modes of operation:
  - input mode, command mode, last-line mode

Let's start with the basics …

vi Editing modes



vi commands

- :w write the current buffer (save)
- :q! quit without saving
- :wq write out (save) and quit
- :x write out (save) and quit
- ZZ write out (save) and quit
- F1 help
- :help
- :help <command>
  - :q to exit help window

vi commands

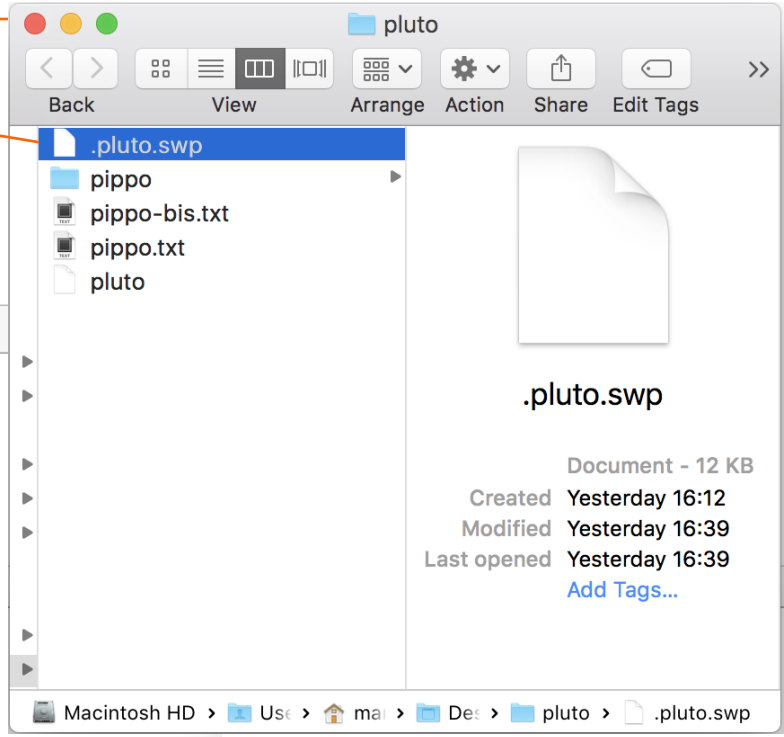
- Insert characters
  - i converts to insert mode
    - then type characters
  - <esc> to exit insert mode
- Delete characters
  - x deletes character under the cursor
- Insert lines
  - o open line below cursor
  - O open line above cursor
  - <esc> to exit insert mode
- Append characters
  - a converts to insert mode after the cursor
  - A converts to insert mode at the end of a line
  - <esc> to exit insert mode

vi commands

- Using motions for movement
  - sx, dx, up, down motions to move
  - <n>w move cursor n words forward
    - 2w move cursor two words forward
    - 3w move cursor four words forward
- Deletion
  - d$ delete until the end of line
  - dw delete until the beginning of next word
  - de delete until the end of current word
  - d + motions (sx, dx, up, down)
  - d<n> + motions
- Using repetition as part of deletion
  - 2dw delete next two words (better to try)
- Deleting a line
  - dd delete line
  - 2dd delete two lines
- Undo
  - u undo one command
  - U restore a line
  - cntl-R redo a command

State files and recovery

Hidden file




```
pluto — vi pluto — 80×24

E325: ATTENTION
Found a swap file by the name ".pluto.swp"
          owned by: marcoautili   dated: Tue Sep 13 16:39:17 2016
         file name: ~marcoautili/Desktop/pluto/pluto
          modified: YES
         user name: marcoautili   host name: iMac-2.local
        process ID: 1129
While opening file "pluto"
             dated: Tue Sep 13 16:07:39 2016

(1) Another program may be editing the same file.
    If this is the case, be careful not to end up with two
    different instances of the same file when making changes.
    Quit, or continue with caution.

(2) An edit session for this file crashed.
    If this is the case, use ":recover" or "vim -r pluto"
    to recover the changes (see ":help recovery").
    If you did this already, delete the swap file ".pluto.swp"
    to avoid this message.

Swap file ".pluto.swp" already exists!
[O]pen Read-Only, (E)dit anyway, (R)ecover, (D)elete it, (Q)uit, (A)bort:
```

Read next slide and try yourself …

More on state files and recovery

- When mentioning a “swap file”, the first thing that comes into your mind might be the kind of file that you would create to increase the swap space on a Unix-like system

- The .swp file in previous slide is not a swap file in the OS sense, it is a state file that keeps your changes since the last save (except the last 200 characters), buffers that you have saved, unsaved macros and the undo structure

- Read more in VIM's help: vim +help\ swap-file

- If there is a crash (power failure, OS crash, etc.), then you can recover your changes using this swap-file. After saving the changes from the swap file to the original file, you will need to exit vim and remove the swap file yourself

http://www.computerworld.com/article/2931534/it-management/what-are-unix-swap-swp-files.html

vi commands

- p put back the deleted text (in new place)
  - one of the delete command above + put = cut-and-paste

- More general cut-and-paste
  - v start visual mode per character
    - move the cursor to highlight and select
  - V start visual mode line wise
  - y yank (copy to buffer then p put in new place
- Search
  - /<phrase> search
  - /<phrase>\c ignore case
  - ?<phrase> search backwards
  - n repeat search
  - N repeat search in the other direction

- Screen movement
  - cntl-b move one screen backward
  - cntl-f move one screen forward
  - cntl-u move half a screen backward
  - cntl-d move half a screen forward

vi commands

- Search for matching parentheses
  - simply put the cursor on (, [, { or },],)
  - % switch between matching left and matching right

- Substitute (replace)
  - :s/thee/the changes first one
  - :s/thee/the/g changes all (global change)
  - :s/thee/the/gc change all with query
  - :#,#s/thee/the/g only change within that line range

- Setting options for search and viewing
  - :set nu enable line numbers
  - :set ic ignore case
  - :set hlsearch highlight matches
  - :set icsearch incremental search
  - :set noic, etc. turn the option off

File navigation commands

- arrow keys move the cursor around
- j, k, h, l move the cursor down, up, left and right (similar to the arrow keys)
- ^ the caret move the cursor to beginning of current line
- $ move the cursor to end of the current line
- <n>G move to the nth line (e.g., 5G moves to 5th line)
- G move to the last line
- gg go to top of file
- w move to the beginning of the next word
- <n>w move forward n word (e.g., 2w moves two words forwards)
- b move to the beginning of the previous word
- <n>b move back n word
- { move backward one paragraph
- } move forward one paragraph
- ctrl-g show position in file

Running Commands

- The vi has the capability to run commands from within the editor
- To run a command, you only need to go into command mode and type :!<any command>
- For example, if you want to check whether a file exists before you try to save your file to that filename, you can type :!ls and you will see the output of ls on the screen

The emacs editor

- Originally started as editor macros in 1976
- Gosling Emacs available for Unix in 1981
- GNU Emacs created by Richard Stallman in 1984
  - very popular editor on Unix until recently
  - history: editor war: emacs vs. vi
- Uses lisp-like macro language for powerful features and extensions:
  - programming language sensitive editing
  - email client
  - news reader
- Has built-in tutorial: ^h-t

The Pico and Nano editors

- Part of the popular pine mail utility on UNIX
  - developed by the University of Washington
- pico = pine email composer
- nano is improved open source of pico available for GNU/Linux
  - very intuitive operation
  - on-screen guide and help

Homework Activities

- Play with all the vi commands
- Search the web for more
- Play with the :help command

Wildcards

- Allow the definition of patterns (also known as globbing patterns or globbing pathnames)
  - a means to deal with a set of files (or directories) at once
- Standard wildcards
  - * match zero or more characters
  - ? match a single character
  - [ ] match a set or a range of characters
    - [agd] - the character is one of those included within the square brackets
    - [a-d] - the character is either a, b, c or d
    - [^a-d] or [!a-d] - reverse a range, the character is none of a, b, c, d
  - { } terms are separated by commas and each term must be the name of something or a wildcard (an "or" relationship, one or the other)
    - cp {*.doc,*.pdf,pippo.txt,mydirectory} - this wildcard will copy anything that matches either wildcard(s) *.doc or *.pdf, or the exact name(s) pippo.txt or mydirectory
    - [a-d] is equivalent to {a,b,c,d}
  - \ (backslash) is used as an "escape" character, i.e., to protect a subsequent special character, thus \\ matches a backslash \
    - Note that you may need to use quotation marks and backslash(es) "\\"

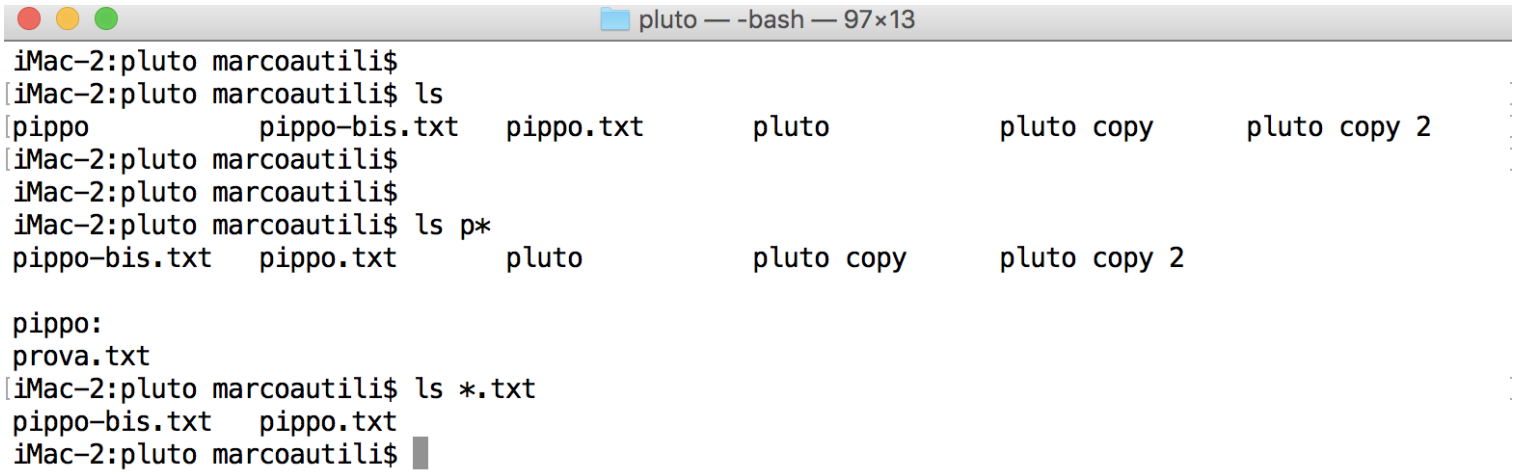
Wildcards

- ls ?ip*
- ls ??pp*
- ls ?ip*.txt
- ls *.???
- ls ?ip*.txt
- ls [psa]*.txt
- ls [0-9] using the hyphen character it ranges over the set {0, 1, 2, ..., 9}
- ls [^a-k]* reverse a range
- ls -lh /home/*/*.txt
- ls -lh ./*/*.txt can be used anywhere within a path

IMPORTANT

wildcards are translated by the shell not by commands

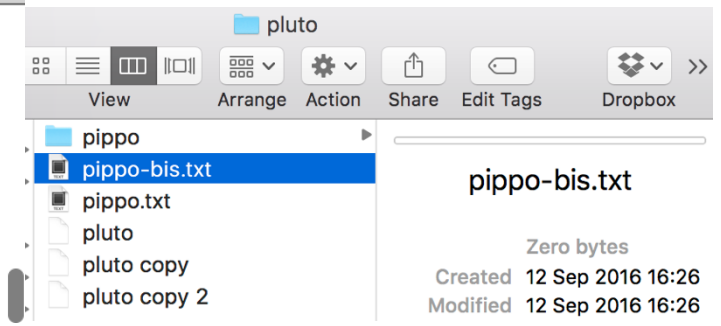
ls *.txt will be translated into ls pippo-bis.txt pippo.txt



Wildcards

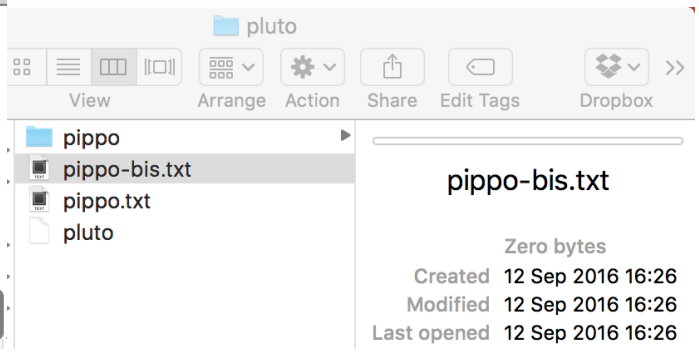
- file ./pluto/*

```
Desktop — -bash — 52×8
iMac-2:Desktop marcoautili$ file ./pluto/*
./pluto/pippo:          directory
./pluto/pippo-bis.txt: empty
./pluto/pippo.txt:      ASCII text
./pluto/pluto:          ASCII text
./pluto/pluto copy:     ASCII text
./pluto/pluto copy 2:   ASCII text
iMac-2:Desktop marcoautili$
```



- mv p*[y2] ./pippo

```
pluto — -bash — 56×9
iMac-2:pluto marcoautili$ ls
pippo           pippo-bis.txt   pippo.txt       pluto
        pluto copy      pluto copy 2
iMac-2:pluto marcoautili$
iMac-2:pluto marcoautili$
iMac-2:pluto marcoautili$ mv p*[y2] ./pippo
iMac-2:pluto marcoautili$ ls
pippo           pippo-bis.txt   pippo.txt       pluto
iMac-2:pluto marcoautili$
```



- etc etc … try yourself ☺

More on globbing

- Long ago, in UNIX V6, there was a program /etc/glob to expand wildcard patterns
- Soon afterward, this became a shell built-in
  - On my Mac, just type man 1 glob to go to the builtin commands page
  - There is also a C Library Routine glob(3) that will perform this function for a user program (#include <glob.h>)

Let's now open a parenthesis on builtin commands…

- According to the standard POSIX.2, 3.13, the rules for wildcard matching are as follows:
  - A string is a wildcard pattern if it contains one of the characters '?', '*' or '['
- That said, globbing can be defined as the operation that expands a wildcard pattern into the list of pathnames matching the pattern

Builtin commands

ppp — less ◂ man builtin — 102×47

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

NAME
     builtin, !, %, ., :, @, {, }, alias, alloc, bg, bind, bindkey, break, breaksw, builtins, case,
     cd, chdir, command, complete, continue, default, dirs, do, done, echo, echotc, elif, else, end,
     endif, endsw, esac, eval, exec, exit, export, false, fc, fg, filetest, fi, for, foreach, getopts,
     glob, goto, hash, hashstat, history, hup, if, jobid, jobs, kill, limit, local, log, login,
     logout, ls-F, nice, nohup, notify, onintr, popd, printenv, pushd, pwd, read, readonly, rehash,
     repeat, return, sched, set, setenv, settc, setty, setvar, shift, source, stop, suspend, switch,
     telltc, test, then, time, times, trap, true, type, ulimit, umask, unalias, uncomplete, unhash,
     unlimit, unset, unsetenv, until, wait, where, which, while -- shell built-in commands

SYNOPSIS
     builtin [-options] [args ...]

DESCRIPTION
     Shell builtin commands are commands that can be executed within the running shell's process.
     Note that, in the case of csh(1) builtin commands, the command is executed in a subshell if it
     occurs as any component of a pipeline except the last.

     If a command specified to the shell contains a slash ``/'', the shell will not execute a builtin
     command, even if the last component of the specified command matches the name of a builtin com-
     mand.  Thus, while specifying ``echo'' causes a builtin command to be executed under shells that
     support the echo builtin command, specifying ``/bin/echo'' or ``./echo'' does not.

     While some builtin commands may exist in more than one shell, their operation may be different
     under each shell which supports them.  Below is a table which lists shell builtin commands, the
     standard shells that support them and whether they exist as standalone utilities.

     Only builtin commands for the csh(1) and sh(1) shells are listed here.  Consult a shell's manual
     page for details on the operation of its builtin commands.  Beware that the sh(1) manual page, at
     least, calls some of these commands ``built-in commands'' and some of them ``reserved words''.
     Users of other shells may need to consult an info(1) page or other sources of documentation.
```

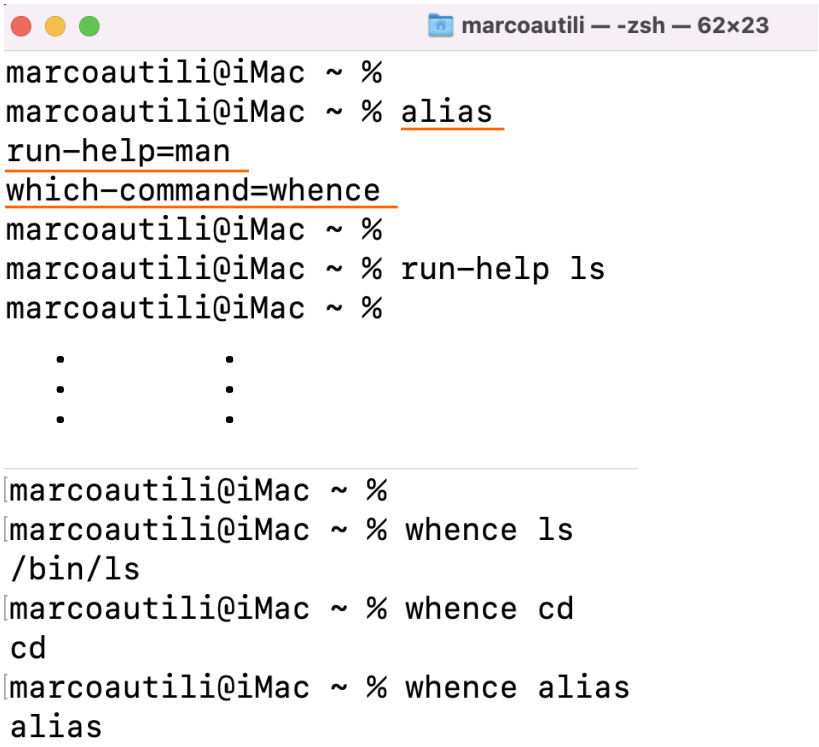
Builtin commands

Commands marked ``No**'' under External do exist externally, but are implemented as scripts using a builtin command of the same name.

Command	External	csh(1)	sh(1)
!	No	No	Yes
%	No	Yes	No
.	No	No	Yes
:	No	Yes	Yes
@	No	Yes	Yes
{	No	No	Yes
}	No	No	Yes
alias	No**	Yes	Yes
alloc	No	Yes	No
bg	No**	Yes	Yes
bind	No	No	Yes
bindkey	No	Yes	No
break	No	Yes	Yes
breaksw	No	Yes	No
builtin	No	No	Yes
builtins	No	Yes	No
case	No	Yes	Yes
cd	No**	Yes	Yes
chdir	No	Yes	Yes
command	No**	No	Yes
complete	No	Yes	No
continue	No	Yes	Yes
default	No	Yes	No
dirs	No	Yes	No
do	No	No	Yes
done	No	No	Yes
echo	Yes	Yes	Yes
echotc	No	Yes	No
elif	No	No	Yes

Note that here I'm running zsh

and these are predefined aliases



man page for builtin commands

man bash (then, search for, e.g., /BUILTIN COMMANDS)

marcoautili — less ‹ man bash — 114×32

```
       event line.  An a may be used as a synonym for g.
   G   Apply the following `s' modifier once to each word in the event line.

SHELL BUILTIN COMMANDS
       Unless otherwise noted, each builtin command documented in this section as accepting options
       preceded by - accepts -- to signify the end of the options.  For example, the :, true, false,
       and test builtins do not accept options.
       : [arguments]
              No effect; the command does nothing beyond expanding arguments and performing any speci-
              fied redirections.  A zero exit code is returned.

        .  filename [arguments]
       source filename [arguments]
              Read  and execute commands from filename in the current shell environment and return the
              exit status of the last command executed from filename.  If filename does not contain  a
              slash,  file names in PATH are used to find the directory containing filename.  The file
              searched for in PATH need not be executable.  When bash is not in posix mode,  the  cur-
              rent directory is searched if no file is found in PATH.  If the sourcepath option to the
              shopt builtin command is turned off, the PATH is not searched.  If  any  arguments  are
              supplied,  they  become  the positional parameters when filename is executed.  Otherwise
              the positional parameters are unchanged.  The return status is the status  of  the  last
              command  exited within the script (0 if no commands are executed), and false if filename
              is not found or cannot be read.

       alias [-p] [name[=value] ...]
              Alias with no arguments or with the -p option prints the list of  aliases  in  the  form
              alias  name=value  on standard output.  When arguments are supplied, an alias is defined
              for each name whose value is given.  A trailing space in  value causes the next word  to
              be  checked  for  alias  substitution  when the alias is expanded.  For each name in the
              argument list for which no value is supplied,  the  name  and  value  of  the  alias  is
              printed.  Alias returns true unless a name is given for which no alias has been defined.
/BUILTIN COMMANDS
```

Builtin commands

- Shell builtins work significantly faster than external programs, because there is no program loading overhead
- However, their code is inherently present in the shell, and thus modifying or updating them requires modifications to the shell
  - for these reason, shell builtins are usually used for simple, almost trivial, functions, such as text output
- Because of the nature of some operating systems, some functions of the systems must necessarily be implemented as shell builtins
  - The most notable example is the cd command, which changes the working directory of the shell program itself
  - Since each executable program runs in a separate process, and working directories are specific to each process, loading cd as an external program would not affect the working directory of the shell that loaded it!
- Use the type or command to find out if given command is an internal builtin or external
  - type -a <command-name>
  - command -V <command-name>

https://en.wikipedia.org/wiki/Shell_builtin

Builtin commands

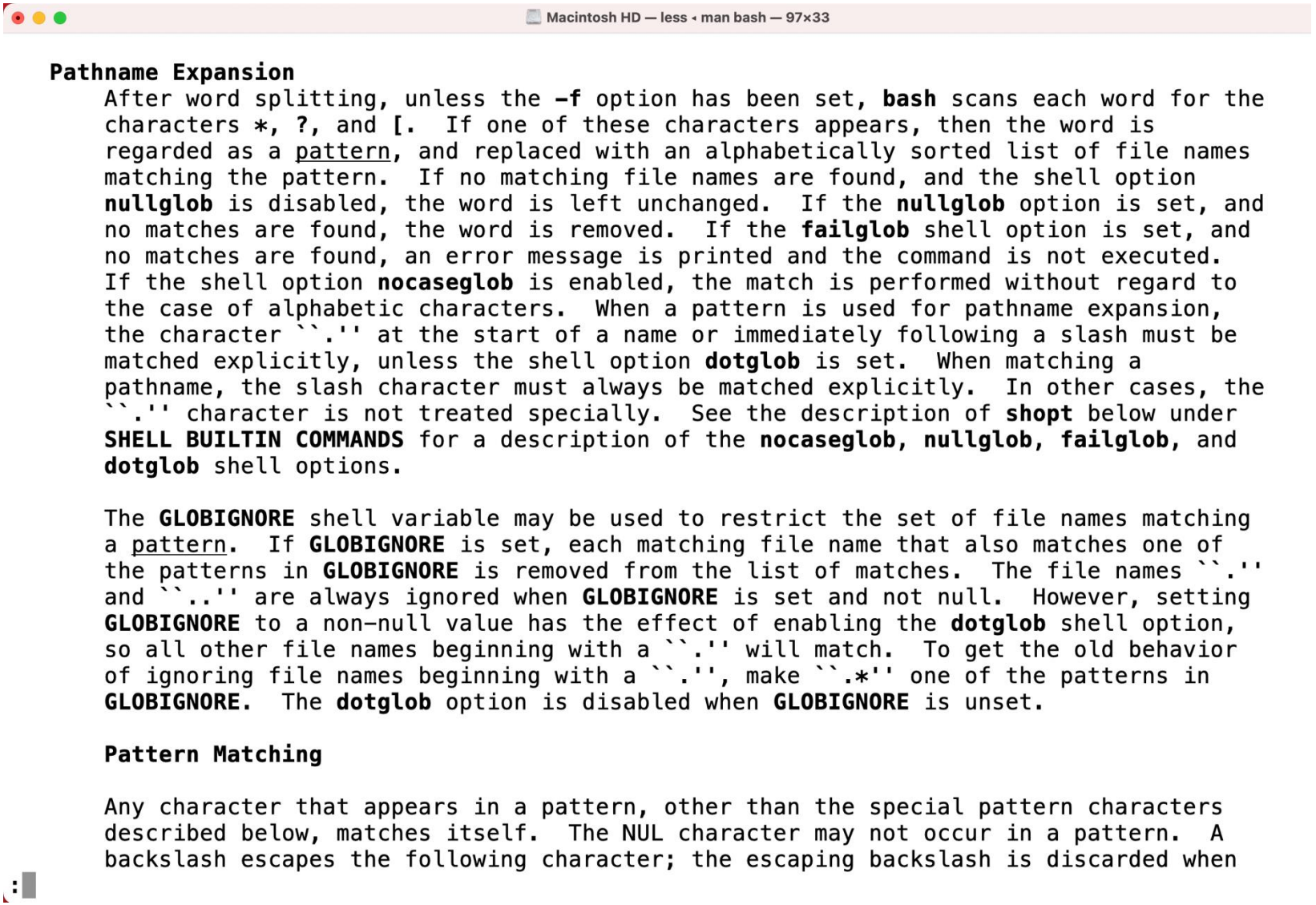
```
marcoautili — -bash —
iMac-di-User:~ marcoautili$
iMac-di-User:~ marcoautili$ type -a top
top is /usr/bin/top
iMac-di-User:~ marcoautili$ type -a alias
alias is a shell builtin
alias is /usr/bin/alias
iMac-di-User:~ marcoautili$ command -V ls
ls is /bin/ls
iMac-di-User:~ marcoautili$ type -a ls
ls is /bin/ls
iMac-di-User:~ marcoautili$ commamd -V date
-bash: commamd: command not found
iMac-di-User:~ marcoautili$ type -a date
date is /bin/date
iMac-di-User:~ marcoautili$ type -a vim
vim is /usr/bin/vim
iMac-di-User:~ marcoautili$ command -V vim
vim is /usr/bin/vim
iMac-di-User:~ marcoautili$
iMac-di-User:~ marcoautili$ man builtin
iMac-di-User:~ marcoautili$
iMac-di-User:~ marcoautili$
iMac-di-User:~ marcoautili$
iMac-di-User:~ marcoautili$
```

top -- display and update sorted information about processes (q to quit)

alias -- see previous slides

```
marcoautili — less • man builtin — 97×51

BUILTIN(1)              BSD General Commands Manual              BUILTIN(1)

NAME
     builtin, !, %, ., :, @, {, }, alias, alloc, bg, bind, bindkey, break, breaksw,
     builtins, case, cd, chdir, command, complete, continue, default, dirs, do, done,
     echo, echotc, elif, else, end, endif, endsw, esac, eval, exec, exit, export,
     false, fc, fg, filetest, fi, for, foreach, getopts, glob, goto, hash, hashstat,
     history, hup, if, jobid, jobs, kill, limit, local, log, login, logout, ls-F, nice,
     nohup, notify, onintr, popd, printenv, pushd, pwd, read, readonly, rehash, repeat,
     return, sched, set, setenv, settc, setty, setvar, shift, source, stop, suspend,
     switch, telltc, test, then, time, times, trap, true, type, ulimit, umask, unalias,
     uncomplete, unhash, unlimit, unset, unsetenv, until, wait, where, which, while --
     shell built-in commands

SYNOPSIS
     builtin [-options] [args ...]

DESCRIPTION
     Shell builtin commands are commands that can be executed within the running
     shell's process.  Note that, in the case of csh(1) builtin commands, the command
     is executed in a subshell if it occurs as any component of a pipeline except the
     last.
```

Back to globbing…



Homework Activities

- Play with wildcards and with different commands

- Define increasingly complex patterns also using them in the middle of paths

- For more information on standard wildcards (globbing pathnames, globbing patterns) refer to the manual page:

  - on my Mac man 1 glob (to go to the builtin commands page)

  - on my Mac man 3 glob (to go to the C Library Functions page)

  - on Linux you should have man 7 glob

    http://man7.org/linux/man-pages/man7/glob.7.html

  - On Mac, for zsh, try also man zshbuiltins

File Permissions

- Allow for assigning access rights to specific users and groups of users
- Control the ability of users to view, change, navigate, and/or execute the contents of the file system
- Unix-like and otherwise POSIX-compliant systems, including Linux-based systems and all macOS versions, have a simple system for managing individual file permissions

File Permissions

- r read - the contents of the file can be viewed
- w write - the contents of the file can be changed
- x execute - the program or script can be executed or run

- owner - a single person who owns the file
  - (typically, the person who created the file, but ownership may be granted to some one else by certain users)
- group - every file belongs to a single group
- others - everyone else who is not in the group or the owner

https://en.wikipedia.org/wiki/File_system_permissions

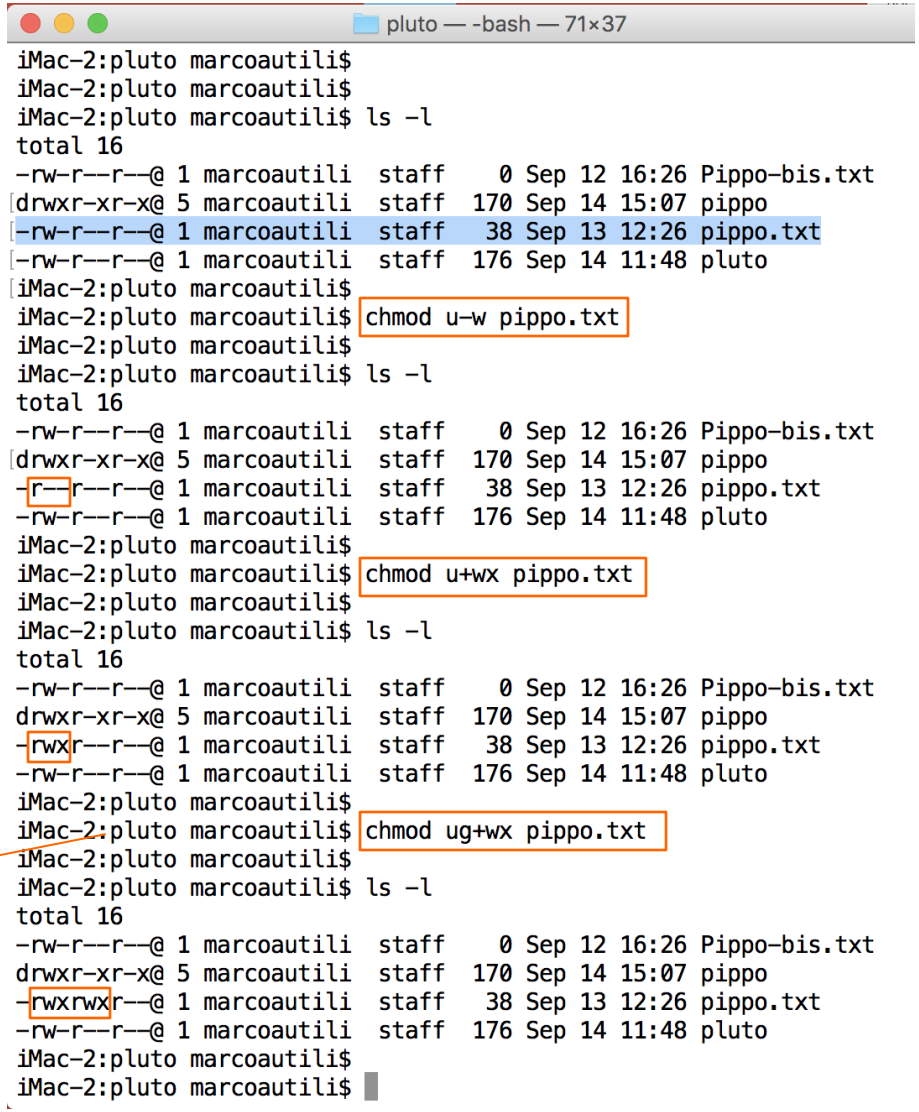
File Permissions

- chmod [permissions] [path]
  chmod stands for change file mode

- Permissions can be changed for
  - user (or owner) -> u
  - group -> g
  - others -> o
  - all -> a

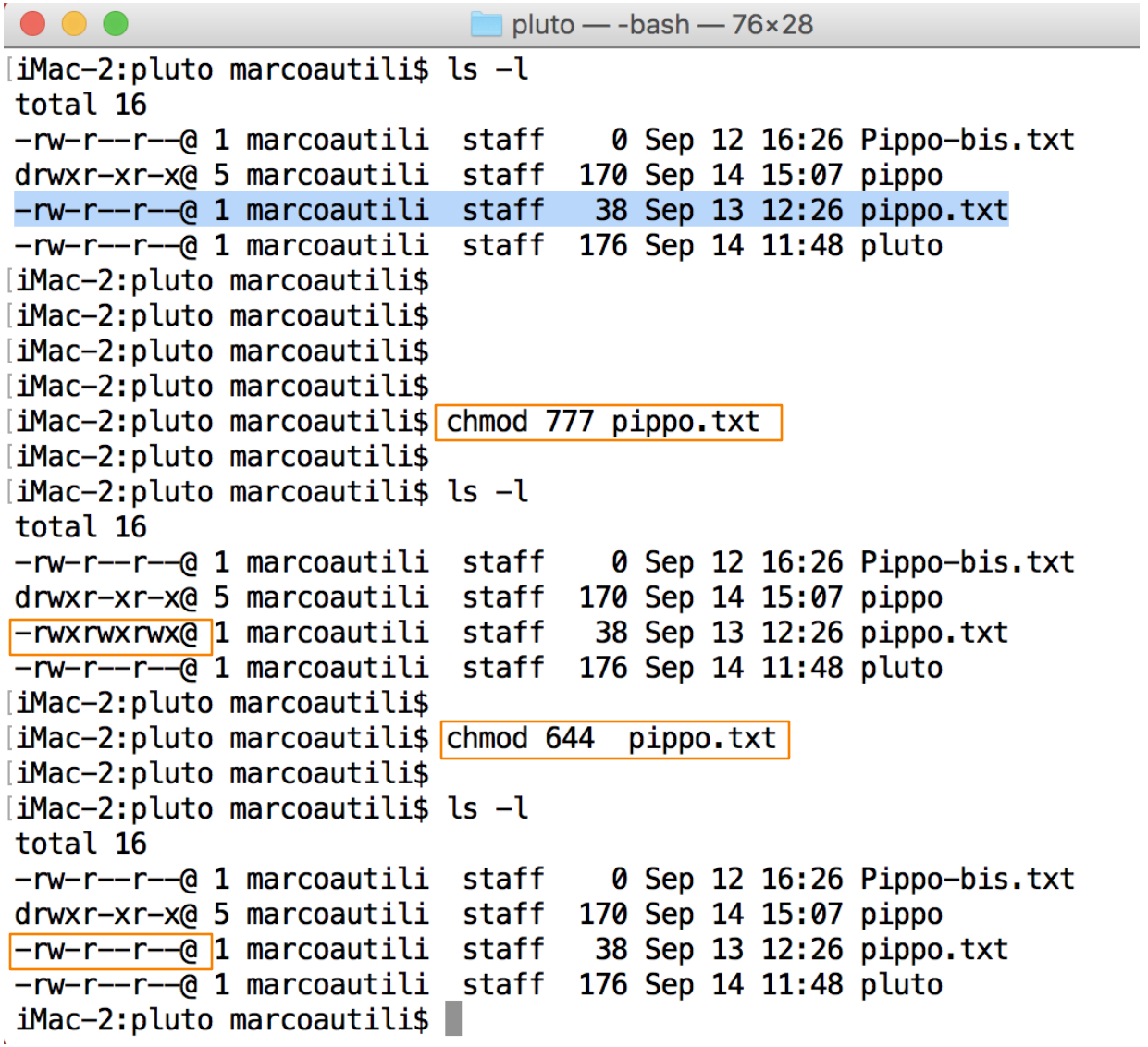
- Permission can be
  - granted -> +
  - revoked -> -

Multiple permissions, e.g., wx, can be assigned at once to, e.g., both user and group ug



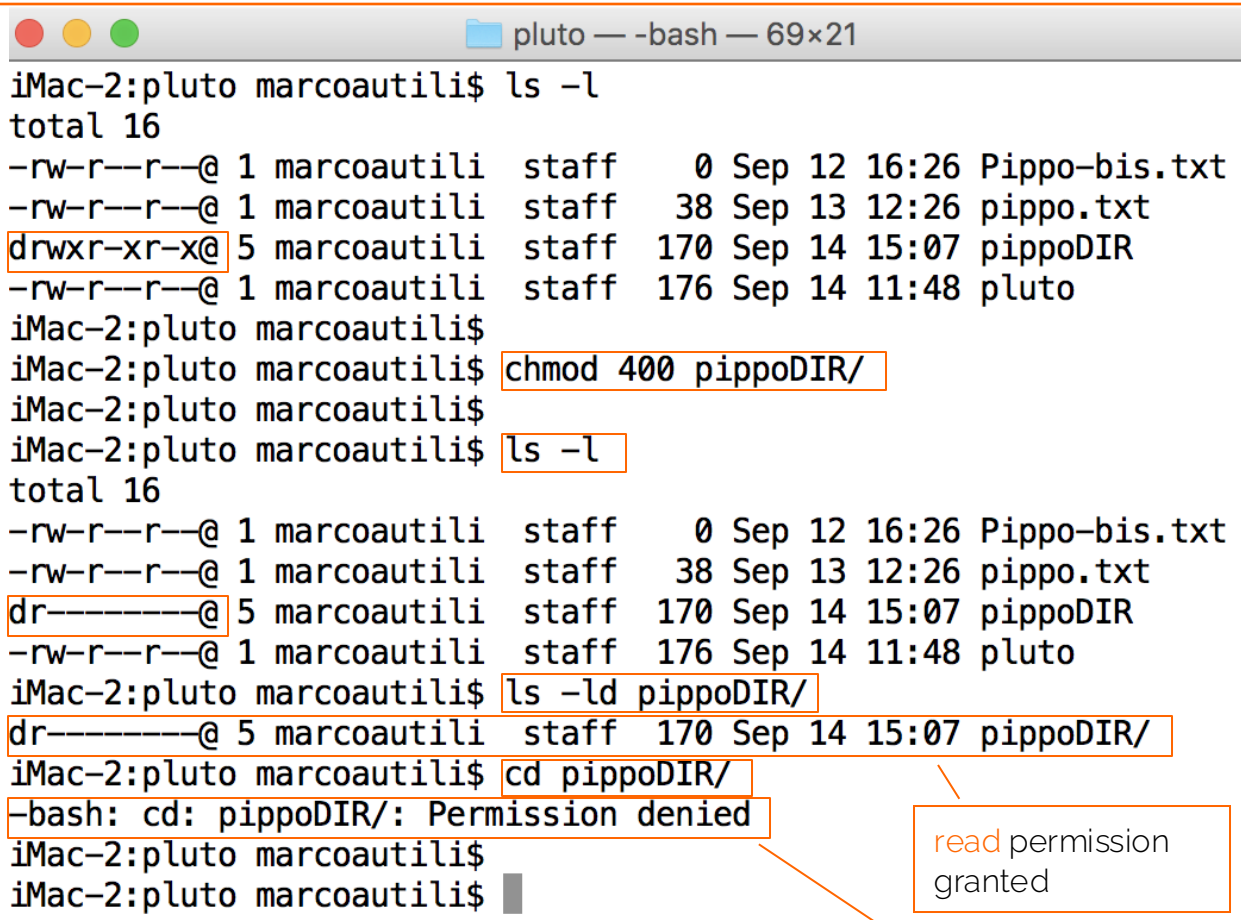
Shorthand to set permissions

Octal	Binary
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1



Directory Permissions

- Different meaning wrt files
- r - the contents of the directory can be read
  - (i.e., the ls command)
- w - the directory can be written
  - (i.e., it is possible to create files and directories in it)
- x - the directory can be entered
  - (i.e., cd)



Note that the -d option (standing for directory) permits to list a directory as a regular file, rather than its content (as usual). Thus, the directory is not searched recursively.

Extended attributes and ACL

Extended attributes (metadata)

- Extended file attributes are file system features that enable users to associate files with metadata not interpreted by the filesystem, whereas regular attributes have a purpose strictly defined by the filesystem (such as permissions or records of creation and modification times)
- Typical uses include storing the author of a document, the character encoding of a plain-text document, or a checksum, cryptographic hash or digital certificate, and discretionary access control information

Access Control List (security)

- An access-control list (ACL), with respect to a computer file system, is a list of permissions attached to an object
- An ACL specifies which users or system processes are granted access to objects, as well as what operations are allowed on given objects
- Each entry in a typical ACL specifies a subject and an operation. For instance, if a file object has an ACL that contains (Alice: read,write; Bob: read), this would give Alice permission to read and write the file and Bob to only read it

https://en.wikipedia.org/wiki/Extended_file_attributes

https://en.wikipedia.org/wiki/Access-control_list

Extended attributes and ACLs

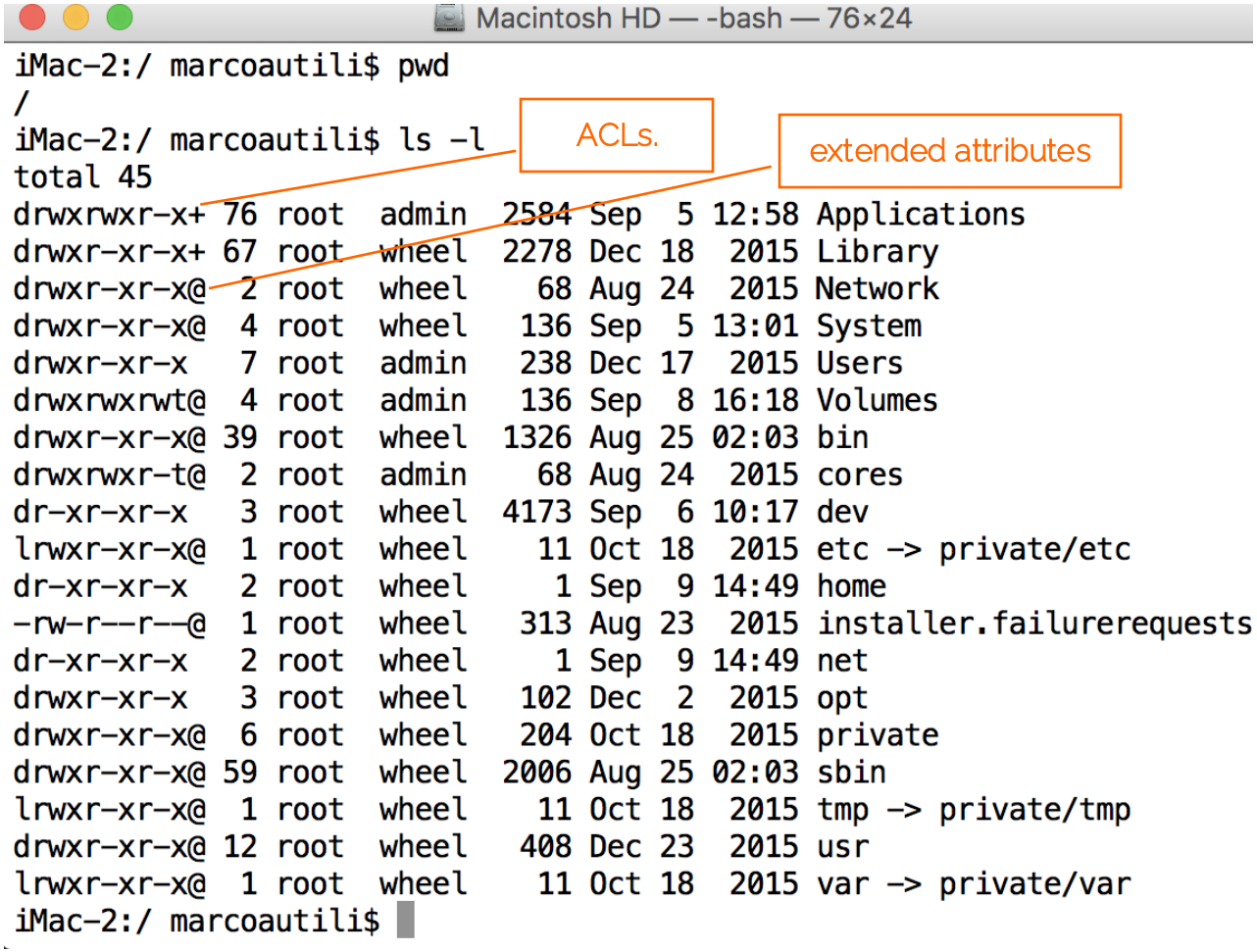
xattr -l <some file>

xattr -c <some file>

Used to display, modify or remove the extended attributes of one or more files, including directories and symbolic links

chmod -N <some file>

Used to remove file ACL(s)

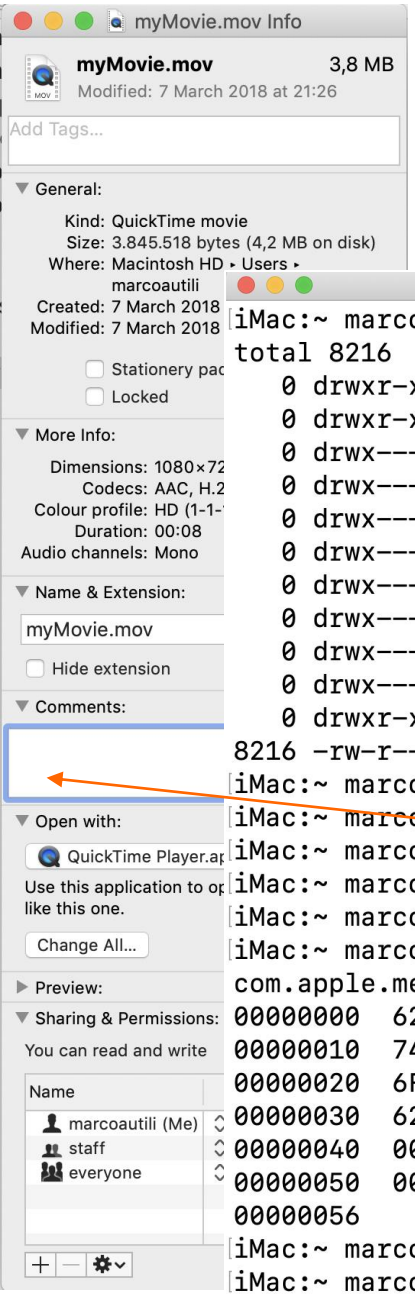


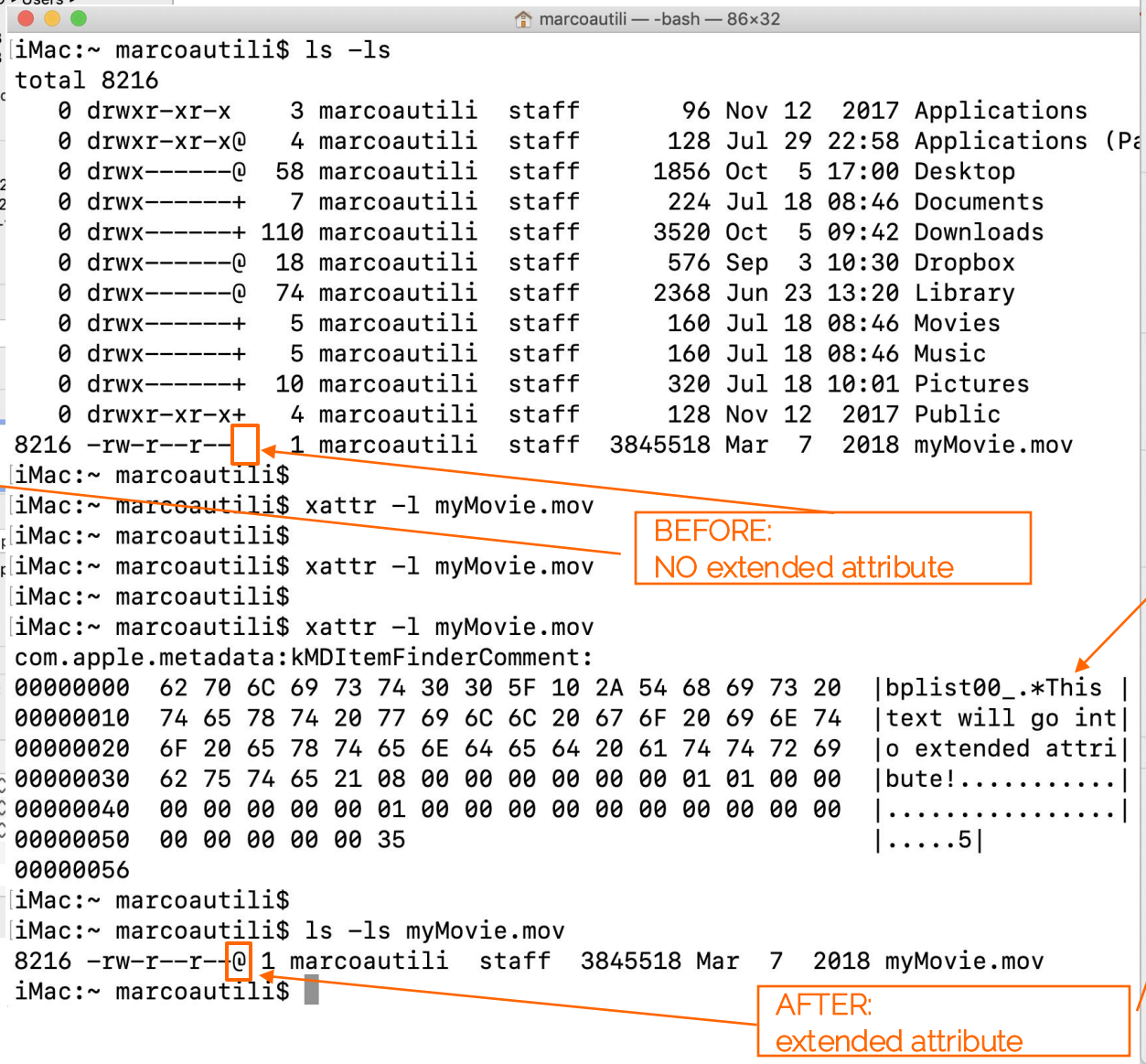
More on that in next slides…

https://en.wikipedia.org/wiki/Extended_file_attributes

https://en.wikipedia.org/wiki/Access-control_list

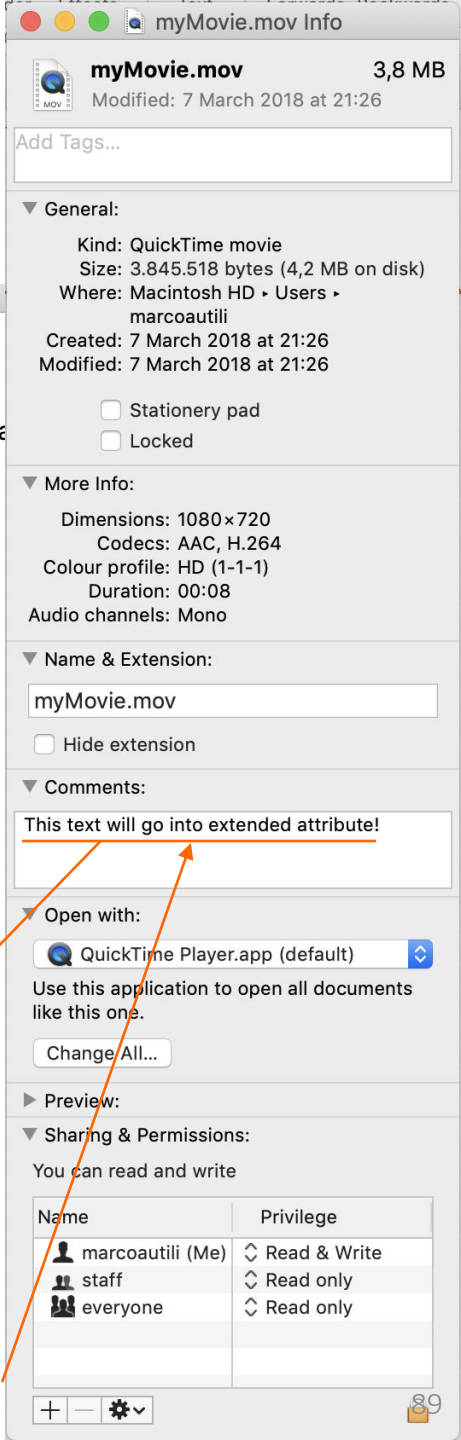
Extended attributes for comments





BEFORE:
NO extended attribute

AFTER:
extended attribute



How to modify file ACLs

```
1 chmod ACL file ...
```

The `chmod` command can be used to modify the Access Control Lists (ACLs) associated with files and directories.

The `chmod` command can also be used to modify the file mode of files and directories: chmod -- change file modes.

If a file/directory has an ACL, the sign "+" will be printed when using the command "ls -l".

Each file/directory has one ACL, containing an ordered list of entries.
Each entry refers to a user or group, and grants ("allow") or denies ("deny") a set of permissions.
In cases where a user and a group exist with the same name, the user or the group name can be prefixed with "user:" or "group:".
If the user or group name contains spaces you can use ':' as the delimiter between name and permission.

http://www.mtitek.com/tutorials/linux/fileperm_chmod_acl.php

-e: print the ACL associated with the file, if present, in long (-l) output

I did not test the following examples on my Mac
Try yourself on your system and adjust if needed

Examples

- Add a new ACL entry that grants the permission "`read`", "`write`", "`append`", and "`execute`" to "user1" on the file "file1".

```
$ ls -le file1
-rwx------ 1 mtitek  mtitek  5  7 Feb 07:45 file1

#change ACL permissions
$ chmod +a "user:user1 allow read,write,append,execute" file1

$ ls -le file1
-rwx------+ 1 mtitek  mtitek  5  7 Feb 07:45 file1
 0: user:user1 allow read,write,execute,append
```

The +a mode parses a new ACL entry from the next argument on the command line and inserts it into the canonical location in the ACL.

If the supplied entry refers to an identity already listed, the two entries are combined.

- Add a new ACL entry that denies the permission "`write`", "`append`", and "`execute`" to "user2" on the file "file1".

```
#change ACL permissions
$ chmod +a "user:user2 deny write,append,execute" file1

$ ls -le
-rwx------+ 1 mtitek  mtitek  5  7 Feb 07:45 file1
 0: user:user2 deny write,execute,append
 1: user:user1 allow read,write,execute,append
```

- Add a new ACL entry, in a specific location, that grants the permission "`read`" to "user3" on the file "file1".

```
#change ACL permissions
$ chmod +a# 2 "user:user3 allow read" file1

$ ls -le
-rwx------+ 1 mtitek  mtitek  5  7 Feb 07:45 file1
 0: user:user2 deny write,execute,append
 1: user:user1 allow read,write,execute,append
 2: user:user3 allow read
```

When a specific ordering is required, the exact location at which an entry will be inserted is specified with the +a# mode.

http://www.mtitek.com/tutorials/linux/fileperm_chmod_acl.php

- Delete an ACL entry on the file "file1".
  - Delete deny write permission for "user2" on the file "file1":

```
#change ACL permissions
$ chmod -a "user:user2 deny write" file1

#Note that only the "deny write" permission is deleted for "user2"
$ ls -le
-rwx------+ 1 mtitek  mtitek  4  7 Feb 07:45 file1
 0: user:user2 deny execute,append
 1: user:user1 allow read,write,execute,append
 2: user:user3 allow read
```

  - Delete deny execute and append permissions for "user2" on the file "file1":

```
$ chmod -a "user:user2 deny execute,append" file1

#Note that "user2" has now no ACL entry
$ ls -le
-rwx------+ 1 mtitek  mtitek  4  7 Feb 07:45 file1
 0: user:user1 allow read,write,execute,append
 1: user:user3 allow read
```

  - Delete an ACL entry by its index:

```
$ chmod -a# 0 file1

#Note that "user1" has now no ACL entry
$ ls -le
-rwx------+ 1 mtitek  mtitek  4  7 Feb 07:45 file1
 0: user:user3 allow read
```

The -a mode is used to delete ACL entries. All entries exactly matching the supplied entry will be deleted.

If the entry lists a subset of rights granted by an entry, only the rights listed are removed. Entries may also be deleted by index using the -a# mode.

http://www.mtitek.com/tutorials/linux/fileperm_chmod_acl.php

- Modify an ACL entry by its index.

```
#change ACL permissions
$ chmod =a# 0 "group:group1 deny write,execute,append" file1

$ ls -le
-rwx------+ 1 mtitek  mtitek  5  7 Feb 07:45 file1
 0: group:group1 deny write,execute,append
```

- Set the permission "`list`", "`search`", "`add_file`", "`add_subdirectory`", and "`delete_child`" to "user1" on the directory "folder1".

```
$ mkdir folder1

$ ls -le
drwxr-xr-x  2 mtitek  mtitek  68  7 Feb 07:52 folder1

#change ACL permissions
$ chmod +a "user:user1 allow list,search,add_file,add_subdirectory,delete_child" folder1

$ ls -le
drwxr-xr-x+ 2 mtitek  mtitek  68  7 Feb 07:52 folder1
 0: user:user1 allow list,add_file,search,add_subdirectory,delete_child
```

See man chmod

http://www.mtitek.com/tutorials/linux/fileperm_chmod_acl.php

Filters, Redirection and Pipelines

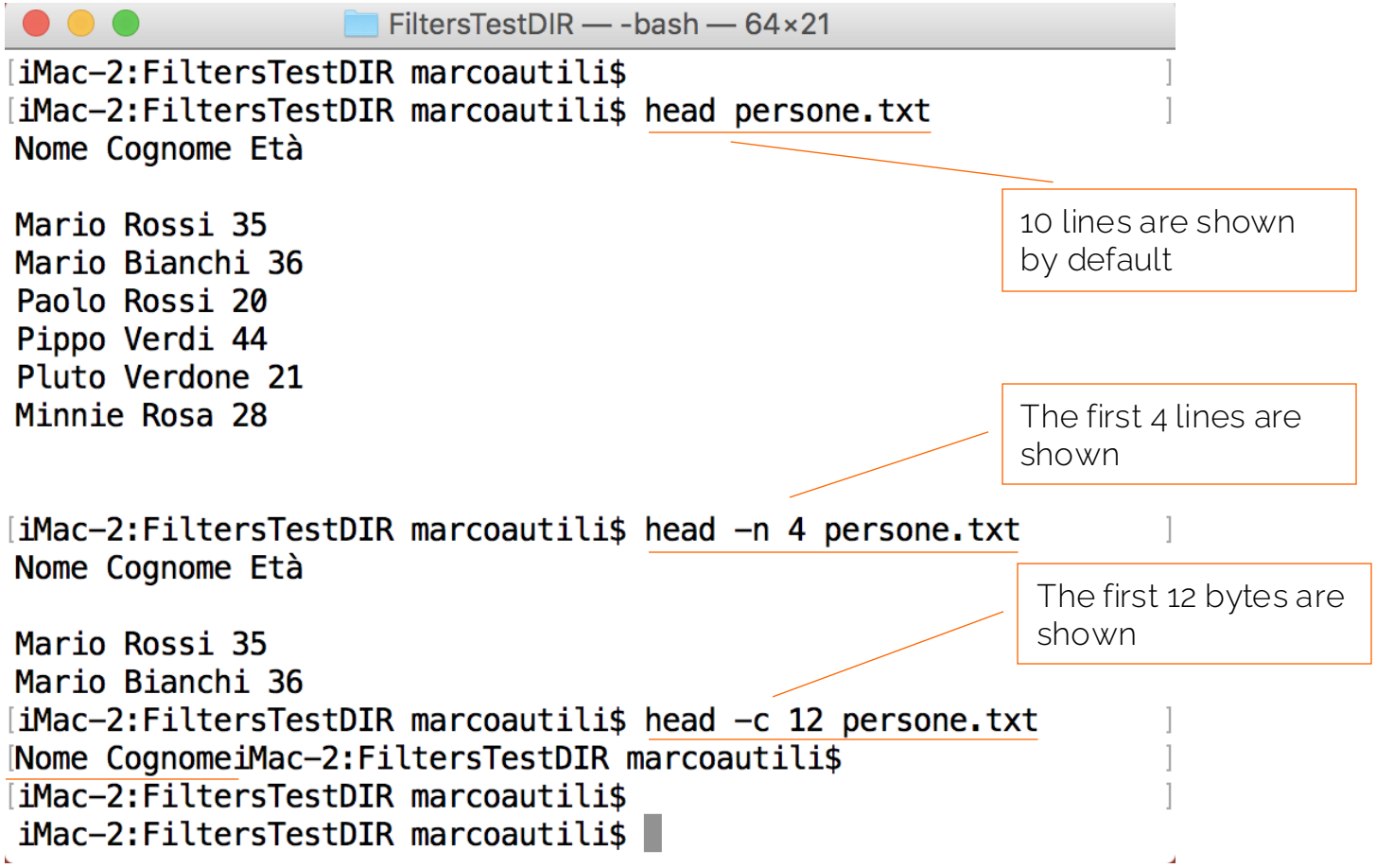
- A filter is a computer program (or subroutine) to process a stream, producing another stream
- In Unix and Unix-like operating systems, a filter is a program that gets its data from its standard input (the main input stream) or files and writes its main results to its standard output (the main output stream)
- Filters do not modify input files
- Filters may be strung together into a pipeline with the pipe operator ("|"). This operator signifies that the main output of the command to the left is passed as main input to the command on the right… more on that later

https://en.wikipedia.org/wiki/Filter_(software)#Unix

Head filter

Print the beginning of a text file (or piped data)

- head [options] <file_name>



Tail filter

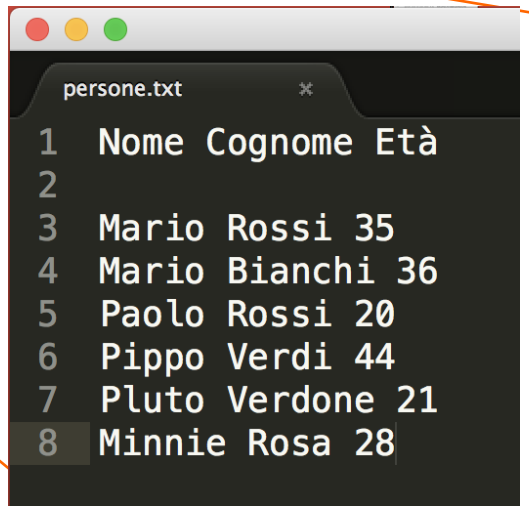
Print the tail of a text file (or piped data)

- tail [options] <file_name>

```
FiltersTestDIR — -bash — 67×10
iMac-2:FiltersTestDIR marcoautili$
iMac-2:FiltersTestDIR marcoautili$
iMac-2:FiltersTestDIR marcoautili$ tail -3 persone.txt
Pippo Verdi 44
Pluto Verdone 21
Minnie Rosa 28
iMac-2:FiltersTestDIR marcoautili$ tail -c 3 persone.txt
28
iMac-2:FiltersTestDIR marcoautili$ tail -c 3 persone.txt
 28iMac-2:FiltersTestDIR marcoautili$
```

No need to write -n 3

There is no End of Line



There is End of Line

Sort filters

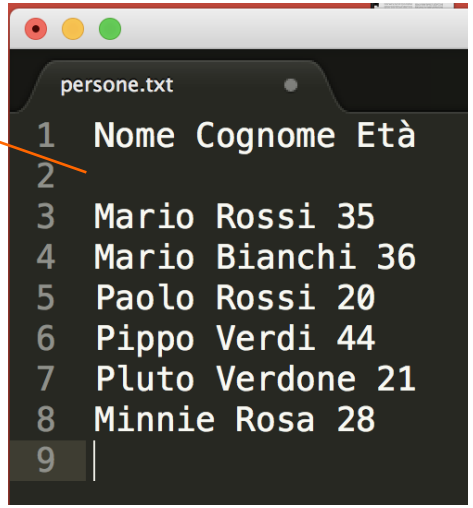
The sort utility sorts text and binary files by lines (by default it will sort alphabetically - see man :-)

- sort [-options] [path]

```
FiltersTestDIR — -bash — 58×13
iMac-2:FiltersTestDIR marcoautili$
iMac-2:FiltersTestDIR marcoautili$
iMac-2:FiltersTestDIR marcoautili$
iMac-2:FiltersTestDIR marcoautili$ sort persone.txt

Mario Bianchi 36
Mario Rossi 35
Minnie Rosa 28
Nome Cognome Età
Paolo Rossi 20
Pippo Verdi 44
Pluto Verdone 21
iMac-2:FiltersTestDIR marcoautili$
```

:(

Empty lines are also sorted!



Number Lines Filter

Add line numbers

- nl [-options] [path]

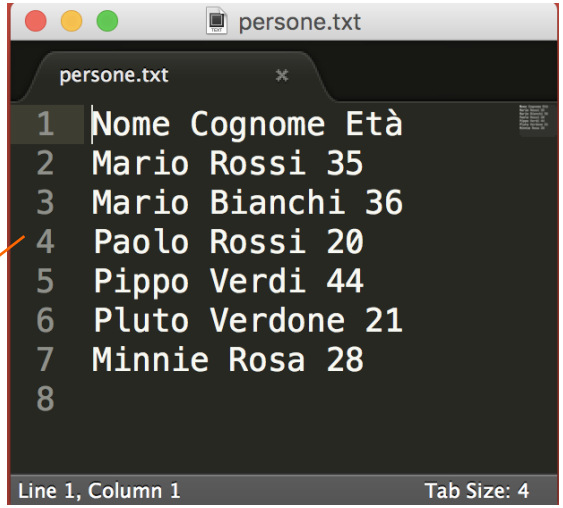
```
FiltersTestDIR — -bash — 65×17
iMac-2:FiltersTestDIR marcoautili$ nl persone.txt
     1  Nome Cognome Età
     2  Mario Rossi 35
     3  Mario Bianchi 36
     4  Paolo Rossi 20
     5  Pippo Verdi 44
     6  Pluto Verdone 21
     7  Minnie Rosa 28
iMac-2:FiltersTestDIR marcoautili$ nl -s '. ' -w 10 persone.txt
         1. Nome Cognome Età
         2. Mario Rossi 35
         3. Mario Bianchi 36
         4. Paolo Rossi 20
         5. Pippo Verdi 44
         6. Pluto Verdone 21
         7. Minnie Rosa 28
iMac-2:FiltersTestDIR marcoautili$
```

Basic formatting

width

Custom formatting

Note that these numbers are not part of the file content!



Word Count Filter

Count lines, words, and characters

- wc [-options] [path]

```
iMac:PART 2 - Filters-Test-DIR marcoautili$
iMac:PART 2 - Filters-Test-DIR marcoautili$ wc persone.txt
       8      21     129 persone.txt
iMac:PART 2 - Filters-Test-DIR marcoautili$ wc -c persone.txt
     129 persone.txt
iMac:PART 2 - Filters-Test-DIR marcoautili$ wc -m persone.txt
     128 persone.txt
iMac:PART 2 - Filters-Test-DIR marcoautili$ wc -l persone.txt
       8 persone.txt
iMac:PART 2 - Filters-Test-DIR marcoautili$ wc -w persone.txt
      21 persone.txt
iMac:PART 2 - Filters-Test-DIR marcoautili$
```

Lines, words, and bytes by default

Bytes only

Chars only

Lines only

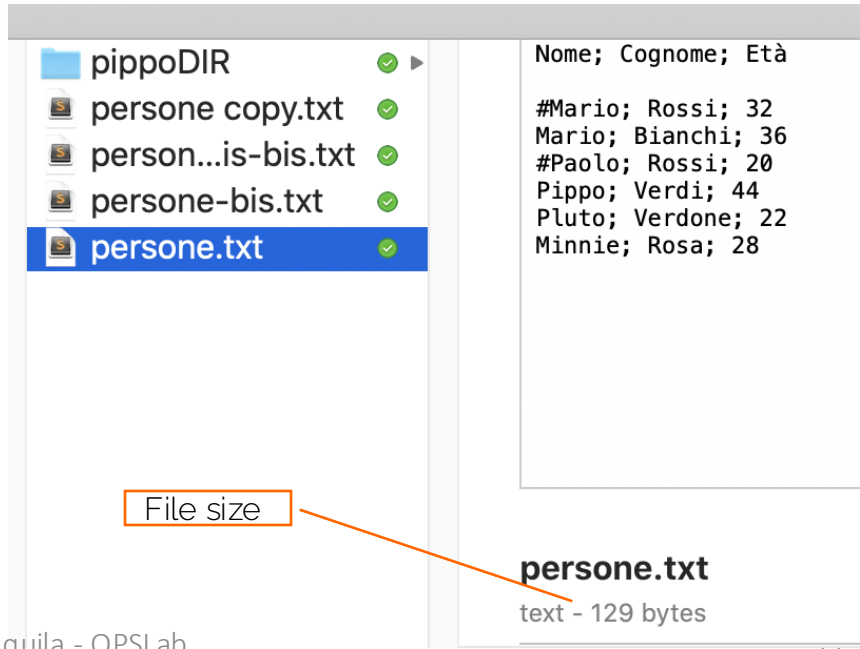
Words only

-c The number of bytes in each input file is written to the standard output. This will cancel out any prior usage of the -m option.

-m The number of characters in each input file is written to the standard output. If the current locale does not support multibyte characters, this is equivalent to the -c option. This will cancel out any prior usage of the -c option.

-l The number of lines in each input file is written to the standard output.

-w The number of words in each input file is written to the standard output.



Cut Filter

Print only certain fields (if the contents of the file is organized into fields)

- cut [-options] [path]

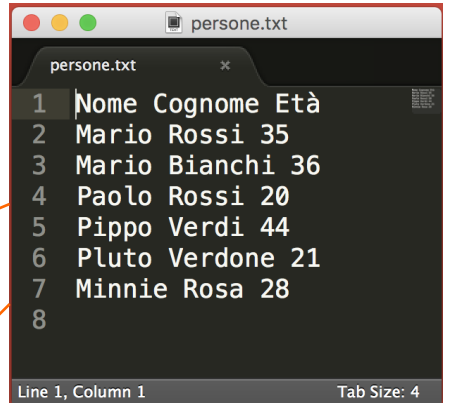
```
FiltersTestDIR — -bash — 65×25
iMac-2:FiltersTestDIR marcoautili$ cut -f 1 -d ' ' persone.txt
Nome
Mario
Mario
Paolo
Pippo
Pluto
Minnie
iMac-2:FiltersTestDIR marcoautili$ cut -f 1 -d ';' persone.txt
Nome Cognome Età
Mario Rossi 35
Mario Bianchi 36
Paolo Rossi 20
Pippo Verdi 44
Pluto Verdone 21
Minnie Rosa 28
iMac-2:FiltersTestDIR marcoautili$ cut -f 1 -d ';' persone.txt
Nome
Mario
Mario
Paolo
Pippo
Pluto
Minnie
iMac-2:FiltersTestDIR marcoautili$
```

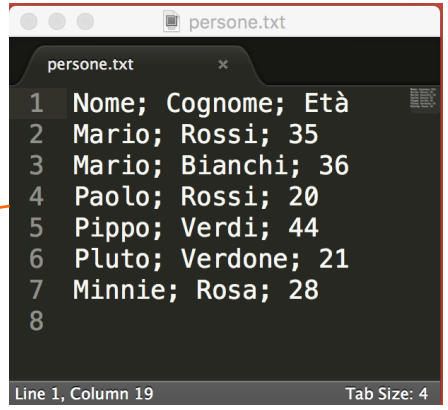
"Single Space" separator. TAB by default

"Semicolon" separator: it does not work on this file!

Try also -f 1,2

"Semicolon" separator





Unique Filter

Remove duplicate adjacent lines

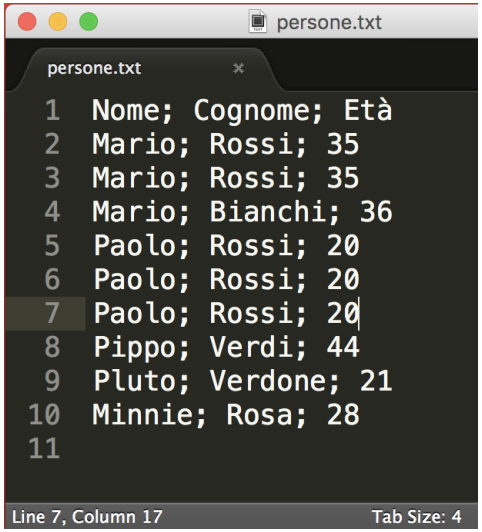
- uniq [options] [path]

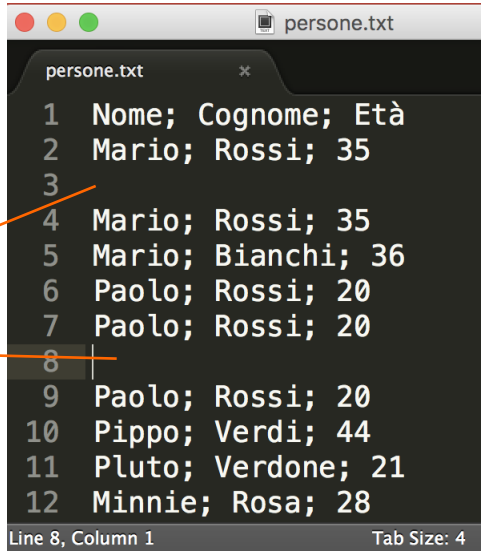
```
FiltersTestDIR — -bash — 53×21
iMac-2:FiltersTestDIR marcoautili$ uniq persone.txt
Nome; Cognome; Età
Mario; Rossi; 35
Mario; Bianchi; 36
Paolo; Rossi; 20
Pippo; Verdi; 44
Pluto; Verdone; 21
Minnie; Rosa; 28
iMac-2:FiltersTestDIR marcoautili$ uniq persone.txt
Nome; Cognome; Età
Mario; Rossi; 35

Mario; Rossi; 35
Mario; Bianchi; 36
Paolo; Rossi; 20

Paolo; Rossi; 20
Pippo; Verdi; 44
Pluto; Verdone; 21
Minnie; Rosa; 28
iMac-2:FiltersTestDIR marcoautili$
```

Adjacent lines: it works!

Not adjacent lines: it does not work!

Later on we will se how to fix this …





Tac Command

Reverse cat: print last line first

- tac [path]

```
FiltersTestDIR — -bash — 53×21
iMac-2:FiltersTestDIR marcoautili$ uniq persone.txt
Nome; Cognome; Età
Mario; Rossi; 35

Mario; Rossi; 35
Mario; Bianchi; 36
Paolo; Rossi; 20

Paolo; Rossi; 20
Pippo; Verdi; 44
Pluto; Verdone; 21
Minnie; Rosa; 28
iMac-2:FiltersTestDIR marcoautili$ tac persone.txt
-bash: tac: command not found
iMac-2:FiltersTestDIR marcoautili$
iMac-2:FiltersTestDIR marcoautili$
iMac-2:FiltersTestDIR marcoautili$ man tac
No manual entry for tac
iMac-2:FiltersTestDIR marcoautili$
iMac-2:FiltersTestDIR marcoautili$
iMac-2:FiltersTestDIR marcoautili$
```

I do not have it!

What about you?...

just try!

Diff Filter

- diff compare two files (line by line) and print the lines that are different

- It outputs a set of instructions for how to change the first file to make it identical to the second file

- It does not modify the files

- It is quite powerful…
  - For instance, the -e option tells diff to output a script, which can be used by the editing programs ed or ex, that contains a sequence of commands
  - … we will make basic use of it as follows

http://www.computerhope.com/unix/udiff.htm

Diff Filter

The first line of the **diff** output will contain:

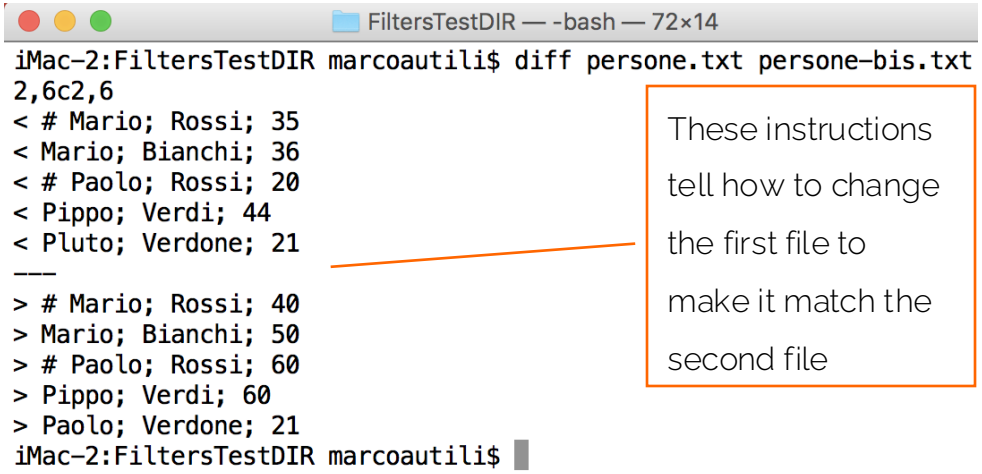
- line numbers corresponding to the first file
- a letter (**a** for *add*, **c** for *change*, or **d** for *delete*)
- line numbers corresponding to the second file.

In the example, "**2,6c2,6**" means:

- "Lines **2** through **6** in the first file need to be **c**hanged to match lines **2** through **6** in the second file."

It then tells us what those lines are in each file:

- Lines preceded by a **<** are lines from the **first file**
- lines preceded by **>** are lines from the **second file**
- the three dashes ("**---**") merely separate the lines of file 1 and file 2.

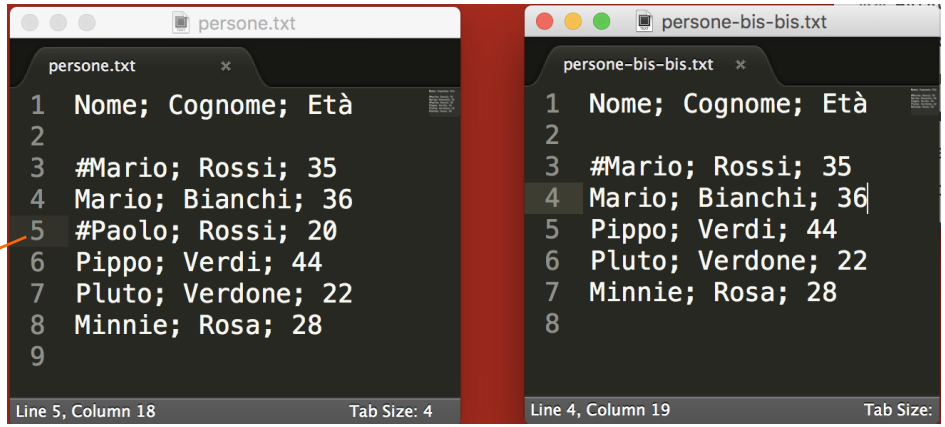


http://www.computerhope.com/unix/udiff.htm

Diff Filter

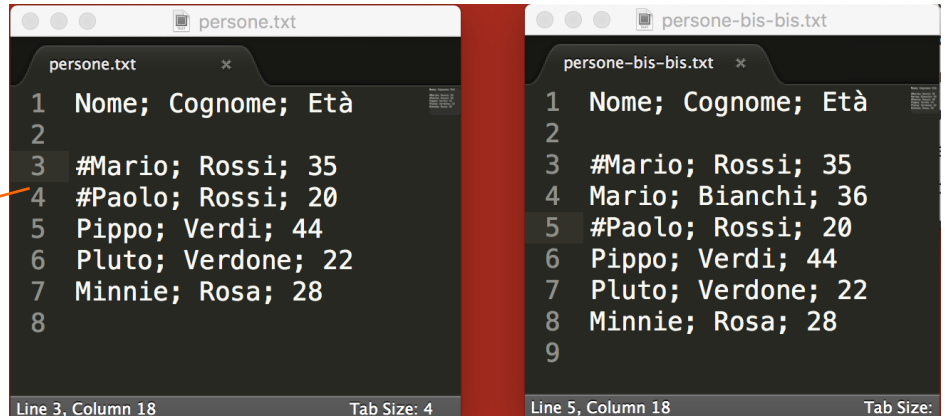
```
Filters-Test-DIR — -bash — 84×10
> Pippo; Verdi; 60
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$ diff persone.txt persone-bis-bis.txt
5d4
< #Paolo; Rossi; 20
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$
```

- You need to delete line 5 in the first file so that both files sync up at line 4
- It then shows us the contents of the line that needs to be deleted.




```
Filters-Test-DIR — -bash — 85×7
desktop-jisqlks:Filters-Test-DIR marcoautili$ diff persone.txt persone-bis-bis.txt
3a4
> Mario; Bianchi; 36
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$
```

- After line 3 in the first file, a line needs to be added: the line to be added is line 4 from the second file
- It then shows us what that line to be added is



Regular Expressions

Regular expressions are a type of globbing pattern used when working with **text**

- Similarly to wildcards for paths, regular expressions allow for creating patterns for text files, although more powerful, e.g., identify all strings in a file that represent URLs, comment lines, email addresses, etc.

IMPORTANT NOTE

- As you know, POSIX (Portable Operating System Interface for uniX) is a collection of standards that define some of the functionality that a (UNIX) operating system should support

- One of these standards defines two flavors of regular expressions:
  - Basic Regular Expressions (BRE)
  - Extended Regular Expressions (ERE)
  - Commands using regular expressions, such as grep and egrep, implement these flavors on POSIX-compliant UNIX systems

Egrep and Regular Expressions

The grep utility searches any given input files, selecting lines that match one or more patterns (egrep stands for extended grep)

- egrep [command line options] <pattern> [path]

```
FiltersTestDIR — -bash — 65×14
iMac-2:FiltersTestDIR marcoautili$ man egrep
iMac-2:FiltersTestDIR marcoautili$
iMac-2:FiltersTestDIR marcoautili$ egrep Mario persone.txt
# Mario; Rossi; 35
Mario; Bianchi; 36
iMac-2:FiltersTestDIR marcoautili$ egrep 'o; V' persone.txt
Pippo; Verdi; 44
Pluto; Verdone; 21
iMac-2:FiltersTestDIR marcoautili$ egrep -n 'o; V' persone.txt
5:Pippo; Verdi; 44
6:Pluto; Verdone; 21
iMac-2:FiltersTestDIR marcoautili$ egrep -c 'o; V' persone.txt
2
iMac-2:FiltersTestDIR marcoautili$
```

```
persone.txt
persone.txt
1  Nome; Cognome; Età
2  # Mario; Rossi; 35
3  Mario; Bianchi; 36
4  # Paolo; Rossi; 20
5  Pippo; Verdi; 44
6  Pluto; Verdone; 21
7  Minnie; Rosa; 28
8
Line 2, Column 3                Tab Size: 4
```

Regular Expressions Format

```
marcoautili — -bash — 91×29
iMac:~ marcoautili$
iMac:~ marcoautili$ man 7 re_format
```

Or simply: man re_format

```
marcoautili — less ◂ man 7 re_format — 96×29

RE_FORMAT(7)         BSD Miscellaneous Information Manual         RE_FORMAT(7)

NAME
     re_format -- POSIX 1003.2 regular expressions

DESCRIPTION
     Regular expressions (``REs''), as defined in IEEE Std 1003.2 (``POSIX.2''), come
     in two forms: modern REs (roughly those of egrep(1); 1003.2 calls these
     ``extended'' REs) and obsolete REs (roughly those of ed(1); 1003.2 ``basic''
     REs).  Obsolete REs mostly exist for backward compatibility in some old programs;
     they will be discussed at the end.  IEEE Std 1003.2 (``POSIX.2'') leaves some
     aspects of RE syntax and semantics open; `=' marks decisions on these aspects
     that may not be fully portable to other IEEE Std 1003.2 (``POSIX.2'') implementa-
     tions.

     A (modern) RE is one= or more non-empty= branches, separated by `|'.  It matches
     anything that matches one of the branches.

     A branch is one= or more pieces, concatenated.  It matches a match for the first,
     followed by a match for the second, etc.

     A piece is an atom possibly followed by a single= `*', `+', `?', or bound.  An
     atom followed by `*' matches a sequence of 0 or more matches of the atom.  An
     atom followed by `+' matches a sequence of 1 or more matches of the atom.  An
     atom followed by `?' matches a sequence of 0 or 1 matches of the atom.
```

Regular Expressions Format

marcoautili — less ◂ man 7 re_format — 95×36

A bound is `{' followed by an unsigned decimal integer, possibly followed by `,' possibly followed by another unsigned decimal integer, always followed by `}'. The integers must lie between 0 and RE_DUP_MAX (255=) inclusive, and if there are two of them, the first may not exceed the second. An atom followed by a bound containing one integer i and no comma matches a sequence of exactly i matches of the atom. An atom followed by a bound containing one integer i and a comma matches a sequence of i or more matches of the atom. An atom followed by a bound containing two integers i and j matches a sequence of i through j (inclusive) matches of the atom.

An atom is a regular expression enclosed in `()' (matching a match for the regular expression), an empty set of `()' (matching the null string)=, a bracket expression (see below), `.' (matching any single character), `^' (matching the null string at the beginning of a line), `$' (matching the null string at the end of a line), a `\' followed by one of the characters `^.[$()|*+?{\' (matching that character taken as an ordinary character), a `\' followed by any other character= (matching that character taken as an ordinary character, as if the `\' had not been present=), or a single character with no other significance (matching that character). A `{' followed by a character other than a digit is an ordinary character, not the beginning of a bound=. It is illegal to end an RE with `\'.

Regular Expressions Format

marcoautili — less ◂ man 7 re_format — 95×36

A bracket expression is a list of characters enclosed in `[]'. It normally matches any single character from the list (but see below). If the list begins with `^', it matches any single character (but see below) not from the rest of the list. If two characters in the list are separated by `-', this is shorthand for the full range of characters between those two (inclusive) in the collating sequence, e.g. `[0-9]' in ASCII matches any decimal digit. It is illegal= for two ranges to share an endpoint, e.g. `a-c-e'. Ranges are very collating-sequence-dependent, and portable programs should avoid relying on them.

To include a literal `]' in the list, make it the first character (following a possible `^'). To include a literal `-', make it the first or last character, or the second endpoint of a range. To use a literal `-' as the first endpoint of a range, enclose it in `[.' and `.]' to make it a collating element (see below). With the exception of these and some combinations using `[' (see next paragraphs), all other special characters, including `\', lose their special significance within a bracket expression.

Within a bracket expression, a collating element (a character, a multi-character sequence that collates as if it were a single character, or a collating-sequence name for either) enclosed in `[.' and `.]' stands for the sequence of characters of that collating element. The sequence is a single element of the bracket expression's list. A bracket expression containing a multi-character collating element can thus match more than one character, e.g. if the collating sequence includes a `ch' collating element, then the RE `[[.ch.]]*c' matches the first five characters of `chchcc'.

……

Regular Expressions made easier for you ☺

A (modern) regular expression is one or more non-empty branches separated by a pipe |
It matches anything that matches one of the branches

A branch is one or more pieces, concatenated. It matches a match for the first, followed by a match for the second, etc.

A piece is an atom possibly followed by a single *, +, ?, or bound

- * - the preceding atom matches 0 or more times
- + - the preceding atom matches 1 or more times
- ? - the preceding atom matches 0 or 1 times only
- {n} - the preceding atom matches exactly n times
- {n,m} –the preceding atom matches at least n and not more than m times (n<=m)
- {n,} –the preceding atom matches at least n times (there is no upper bound)

  - A single { or a { followed by a character other than a digit is an ordinary character, not the beginning of a bound. Thus, e.g., a single { matches { and {a matches {a
  - It is illegal to end a regular expression with \

Regular Expressions made easier for you ☺

An atom is

- (reg exp) - a regular expression enclosed in () matches a match for the regular expression
- () – matches the empty string
- . - a dot matches any single character
- ^ - matches the null string at the beginning of a line
- $ - matches the null string at the end of a line
- \ followed by one of the characters ^ . [$ ( ) | * + ? { \ - matches that character taken as an ordinary character
- \ followed by any other character - matches that character taken as an ordinary character, as if the \ had not been present
- A single character with no other significance - matches that character

Regular Expressions made easier for you ☺

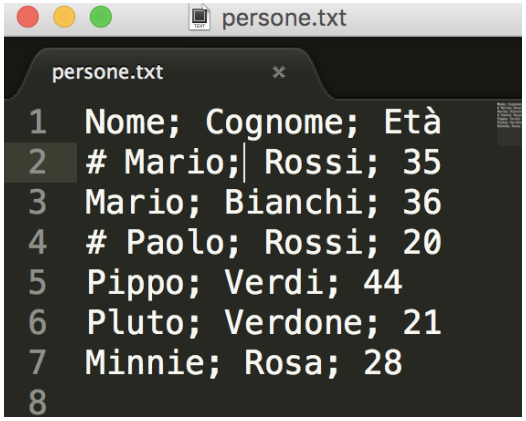
An atom is also a bracket expression

A bracket expression is a list of characters enclosed in []

- [agd] – matches a single character included within []
- [^agd] – matches any single character different from those included within []
- [c-f] - the dash within the square brackets operates as a range (inclusive). In this case it means either the letters c, d, e or f

  - It is illegal for two ranges to share an endpoint, e.g., [a-c-e]
    However, [h-j4-9] is legal as well as [a-dh-j] and [5-81-3]
  - To include the character] in the list, make it the first character of the list, e.g., []abh9] or []]
  - Note that [] can be used to escape any of the characters ^ . [$ ( ) | * + ? { \ , e.g., [$], [?], [*]
    Thus, [$], [?], [*] are equivalent to the escaping \$, \?, \ *
  - Note that []]] is equivalent to]] and indeed it matches]] since the bracket expression []] matches] and the single] matches]

Egrep: simple examples




```
FiltersTestDIR — -bash — 76×14
iMac-2:FiltersTestDIR marcoautili$ egrep '35|36' persone.txt
# Mario; Rossi; 35
Mario; Bianchi; 36
iMac-2:FiltersTestDIR marcoautili$ egrep 8$ persone.txt
Minnie; Rosa; 28
iMac-2:FiltersTestDIR marcoautili$ egrep '[aeiou]{2,}' persone.txt
# Mario; Rossi; 35
Mario; Bianchi; 36
# Paolo; Rossi; 20
Minnie; Rosa; 28
iMac-2:FiltersTestDIR marcoautili$ egrep '3.' persone.txt
# Mario; Rossi; 35
Mario; Bianchi; 36
iMac-2:FiltersTestDIR marcoautili$
```

Persons 35 or 36 years old

Lines ending with 8

Lines containing at least two adjacent vowels

Persons 35 or 36 years old (as before)

https://www.regular-expressions.info/posix.html

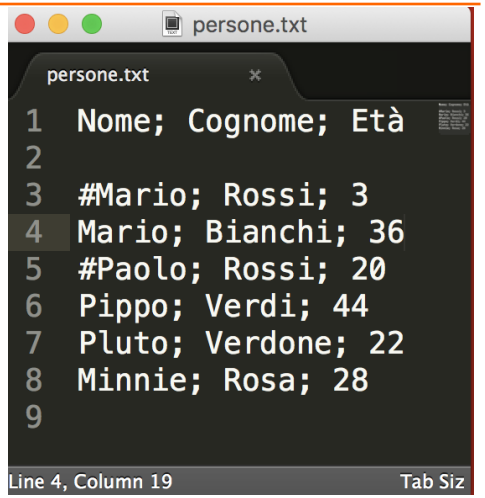
Egrep: simple examples

```
Filters-Test-DIR — -bash — 82×14
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$ egrep '(ss)|(pp)' persone.txt
#Mario; Rossi; 3
#Paolo; Rossi; 20
Pippo; Verdi; 44
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$ egrep '^(Ma)' persone-bis-bis.txt
Mario; Bianchi; 36
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$ egrep '(ss) | (pp)' persone.txt
desktop-jisqlks:Filters-Test-DIR marcoautili$
```

All the lines starting with "Ma"

All the lines containing "ss" or "pp"

No space in between!

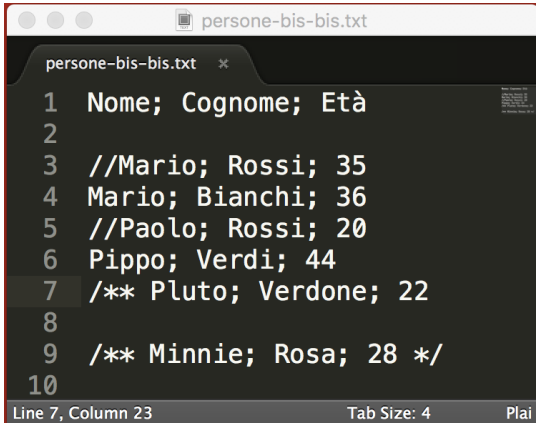



```
Filters-Test-DIR — -bash — 98×6
desktop-jisqlks:Filters-Test-DIR marcoautili$
desktop-jisqlks:Filters-Test-DIR marcoautili$ egrep '^(/\*\*).*(\*/)$|^(//)' persone-bis-bis.txt
//Mario; Rossi; 35
//Paolo; Rossi; 20
/** Minnie; Rosa; 28 */
desktop-jisqlks:Filters-Test-DIR marcoautili$
```

All the lines:
- starting with "/**" and ending with "*/"

OR
- starting with "//"

Escape: "\" in front of a key character, removes its special meaning

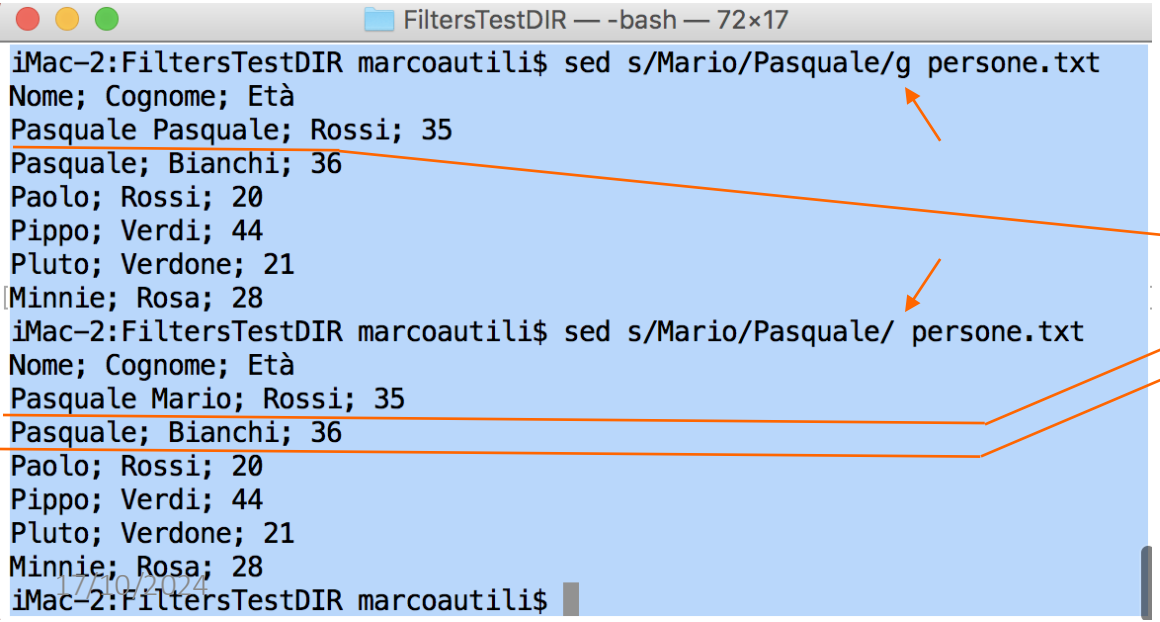


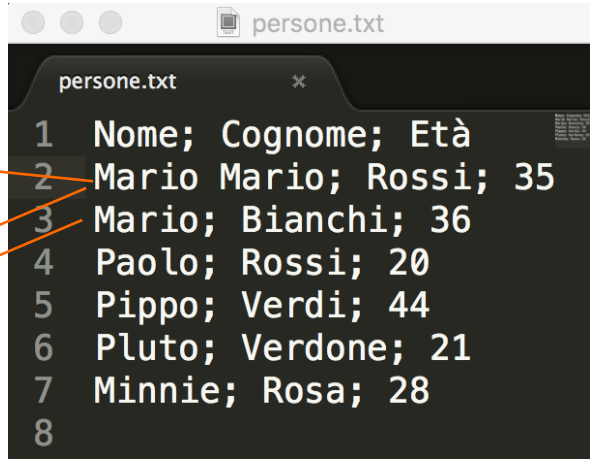
Stream editor Filter

- Very powerful… some basics follow
- Type CTRL+c in case of undesired behavior ☺

  sed <command> [file]

- Search and replace
  - the basic expression format is s/search/replace/g:
    - "s" stands for substitute and "g" stands for global (it is optional). There are others …
    - s/search/replace -> will replace only the first instance on each line



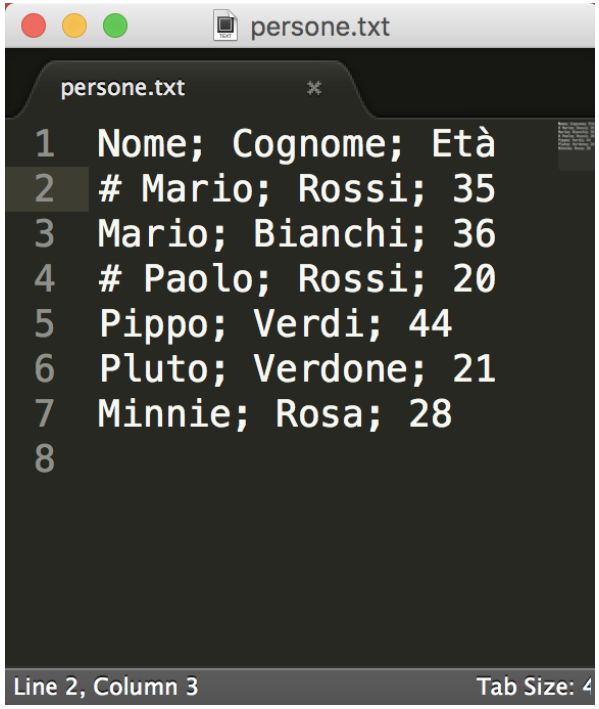


Stream Editor Filter

```
FiltersTestDIR — -bash — 67×14
iMac-2:FiltersTestDIR marcoautili$
iMac-2:FiltersTestDIR marcoautili$ sed -e '1,2d' persone.txt
Mario; Bianchi; 36
# Paolo; Rossi; 20
Pippo; Verdi; 44
Pluto; Verdone; 21
Minnie; Rosa; 28
iMac-2:FiltersTestDIR marcoautili$ sed -e '/^#/d' persone.txt
Nome; Cognome; Età
Mario; Bianchi; 36
Pippo; Verdi; 44
Pluto; Verdone; 21
Minnie; Rosa; 28
iMac-2:FiltersTestDIR marcoautili$
```

Delete the first two lines

Delete #-commented lines



One method of combining multiple commands is to use a -e before each command:
sed -e 's/a/A/' -e 's/b/B/' persone.txt

The "-e" isn't needed in the case of a single command, e.g., sed 's/a/A/' persone.txt

The long argument version should be:
'sed --expression='s/a/A/' --expression='s/b/B/ (NOT WORKING ON MY macOS... Try on your machine)

Sed and regular expression

marcoautili — less ◂ man sed — 86×27

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

NAME
     sed -- stream editor

SYNOPSIS
     sed [-Ealn] command [file ...]
     sed [-Ealn] [-e command] [-f command_file] [-i extension] [file ...]

DESCRIPTION
     The sed utility reads the specified files, or the standard input if no
     files are specified, modifying the input as specified by a list of com-
     mands.  The input is then written to the standard output.

     A single command may be specified as the first argument to sed.  Multiple
     commands may be specified by using the -e or -f options.  All commands
     are applied to the input in the order they are specified regardless of
     their origin.

     The following options are available:

     -E      Interpret regular expressions as extended (modern) regular
             expressions rather than basic regular expressions (BRE's).  The
             re_format(7) manual page fully describes both formats.

:
```

Example in next slide

Sed: BRE VS ERE

es2.txt

```
[aaca]]
[a2cbb]
[a2bc]
```

src-solution — bash — 43×12

```
bash-3.2$
bash-3.2$ sed -e /'^\[a{2,}'/d es2.txt
[aaca]]
[a2cbb]
[a2bc]
bash-3.2$
bash-3.2$
bash-3.2$ sed -E /'^\[a{2,}'/d es2.txt
[a2cbb]
[a2bc]
bash-3.2$
bash-3.2$
```

BRE

ERE

Awk Filter

- AWK is a language for processing text files
- A file is treated as a sequence of records, and by default each line is a record
- Each line is broken up into a sequence of fields, so we can think of the first word in a line as the first field, the second word as the second field, and so on
- AWK reads the input a line at a time
- An AWK program is a sequence of pattern-action statements
  - A line is scanned for each pattern in the program, and for each pattern that matches, the associated action is executed

https://en.wikipedia.org/wiki/Filter_(software)#Unix

Awk Filter

- Special-purpose language for line-oriented pattern processing

  pattern {action}

- action =
  - if (conditional) *statement* else *statement*
  - while (conditional) statement
  - break
  - continue
  - variable=expression
  - print expression-list

We will not treat AWK

* https://en.wikipedia.org/wiki/Filter_(software)#Unix

Homework activities

- Play with all the filters
- Have a look at man egrep and play with its options
- Define increasingly complex regular expressions and test them using egrep on a file you have aptly defined

Pipeline and Redirection

Let's introduce standard data streams first

- Standard data streams are pre-connected communication channels
  - Every program/command executed on the command line automatically has the following standard streams connected to it

STDIN (0)

- Standard input (data fed into the program, from the terminal through the keyboard by default)

STDOUT (1)

- Standard output (data printed by the program, defaults to the terminal)

STDERR (2)

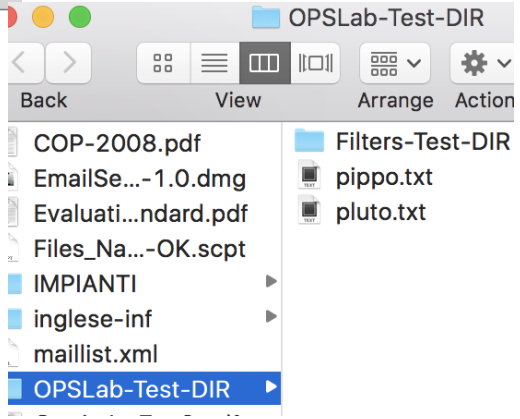
- Standard error (for error messages, also defaults to the terminal)

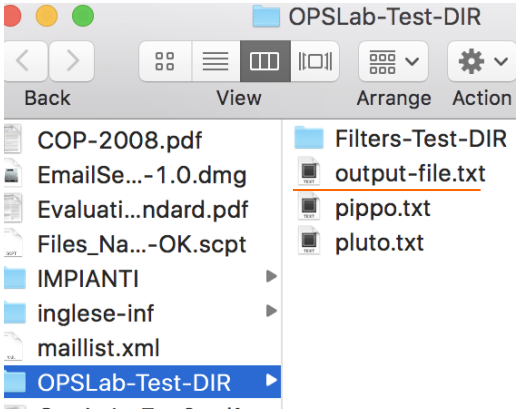
The three streams have numbers associated with them

Redirecting STDOUT to a file

The operator ">" (or ">>") permits to redirect the output of a program or a command to a file, rather than printing the output on the screen (as by default)

```
OPSLab-Test-DIR — -bash — 81×42
iMac-2:OPSLab-Test-DIR marcoautili$ pwd
/Users/marcoautili/Desktop/OPSLab-Test-DIR
iMac-2:OPSLab-Test-DIR marcoautili$
iMac-2:OPSLab-Test-DIR marcoautili$ ls -la > output-file.txt
iMac-2:OPSLab-Test-DIR marcoautili$
iMac-2:OPSLab-Test-DIR marcoautili$ cat output-file.txt
total 32
drwxr-xr-x@  7 marcoautili  staff   238 Sep 20 15:32 .
drwx------@ 34 marcoautili  staff  1156 Sep 20 15:32 ..
-rw-r--r--@  1 marcoautili  staff  6148 Sep 20 15:30 .DS_Store
drwxr-xr-x@  6 marcoautili  staff   204 Sep 20 08:53 Filters-Test-DIR
-rw-r--r--   1 marcoautili  staff     0 Sep 20 15:32 output-file.txt
-rw-r--r--@  1 marcoautili  staff    14 Sep 20 15:26 pippo.txt
-rw-r--r--@  1 marcoautili  staff    14 Sep 20 15:29 pluto.txt
iMac-2:OPSLab-Test-DIR marcoautili$
iMac-2:OPSLab-Test-DIR marcoautili$ diff pippo.txt pluto.txt >> output-file.txt
iMac-2:OPSLab-Test-DIR marcoautili$
iMac-2:OPSLab-Test-DIR marcoautili$ cat output-file.txt
total 32
drwxr-xr-x@  7 marcoautili  staff   238 Sep 20 15:32 .
drwx------@ 34 marcoautili  staff  1156 Sep 20 15:32 ..
-rw-r--r--@  1 marcoautili  staff  6148 Sep 20 15:30 .DS_Store
drwxr-xr-x@  6 marcoautili  staff   204 Sep 20 08:53 Filters-Test-DIR
-rw-r--r--   1 marcoautili  staff     0 Sep 20 15:32 output-file.txt
-rw-r--r--@  1 marcoautili  staff    14 Sep 20 15:26 pippo.txt
-rw-r--r--@  1 marcoautili  staff    14 Sep 20 15:29 pluto.txt
1c1
< pippo-file.txt
\ No newline at end of file
---
> pluto-file.txt
\ No newline at end of file
iMac-2:OPSLab-Test-DIR marcoautili$ diff pippo.txt pluto.txt > output-file.txt
iMac-2:OPSLab-Test-DIR marcoautili$
iMac-2:OPSLab-Test-DIR marcoautili$ cat output-file.txt
1c1
< pippo-file.txt
\ No newline at end of file
---
> pluto-file.txt
\ No newline at end of file
iMac-2:OPSLab-Test-DIR marcoautili$
```



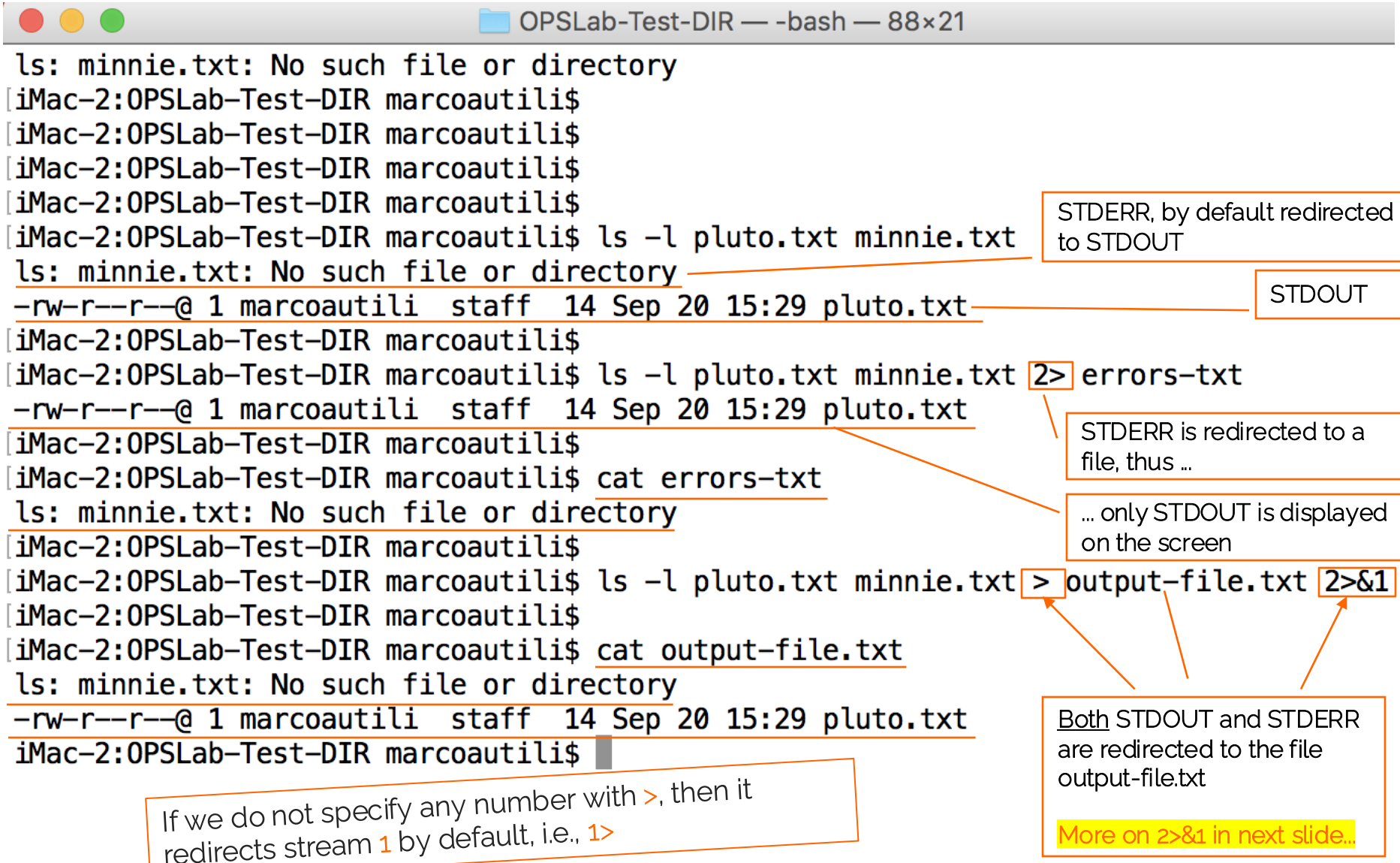


Redirecting STDIN/STDOUT from/to a file

```
OPSLab-Test-DIR — -bash — 85×16
iMac-2:OPSLab-Test-DIR marcoautili$
iMac-2:OPSLab-Test-DIR marcoautili$ cat output-file.txt
1c1
< pippo-file.txt
\ No newline at end of file
---
> pluto-file.txt
\ No newline at end of file
iMac-2:OPSLab-Test-DIR marcoautili$ wc -l < output-file.txt
       6
iMac-2:OPSLab-Test-DIR marcoautili$
iMac-2:OPSLab-Test-DIR marcoautili$ wc -l < output-file.txt > output-file-bis.txt
iMac-2:OPSLab-Test-DIR marcoautili$
iMac-2:OPSLab-Test-DIR marcoautili$ cat output-file-bis.txt
       6
iMac-2:OPSLab-Test-DIR marcoautili$
```

In PART 3, we will see another type of redirection, called Here Strings <<<

Redirecting STDERR



Duplicating File Descriptors

The redirection operator

[n]<&word

is used to duplicate **input** file descriptors

- If word expands to one or more digits, the file descriptor denoted by n is made to be a copy of that file descriptor. If the digits in word do not specify a file descriptor open for input, a redirection error occurs.
- If word evaluates to '-', file descriptor n is closed. If n is not specified, the standard input (file descriptor 0) is used

The operator

[n]>&word

is used similarly to duplicate **output** file descriptors

- If n is not specified, the standard output (file descriptor 1) is used. If the digits in word do not specify a file descriptor open for output, a redirection error occurs. If word evaluates to '-', file descriptor n is closed. As a special case, if n is omitted, and word does not expand to one or more digits or '-', the standard output and standard error are redirected as described previously

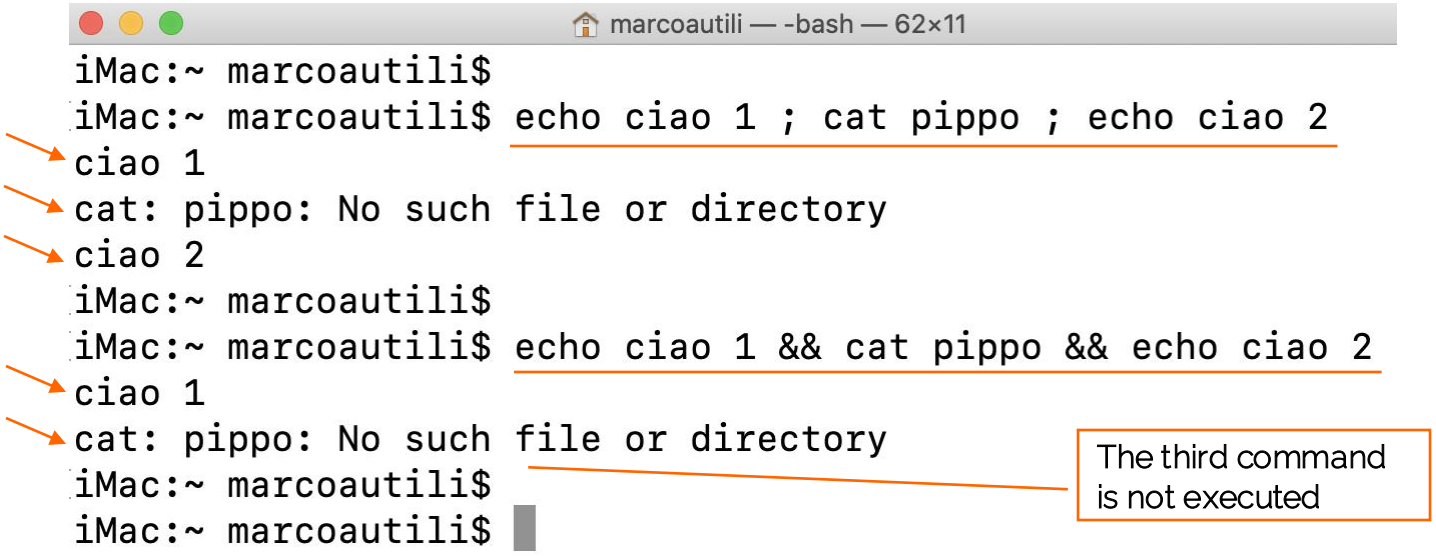
Run commands in sequence

Semicolons ";"

- run each command, regardless if the preceding ones fail
- $ com1 ; com2 ; com3

Double-ampersands "&&"

- run the next command if the preceding command succeeded, and all the commands correctly set exit codes
- $ com1 && com2 && com3

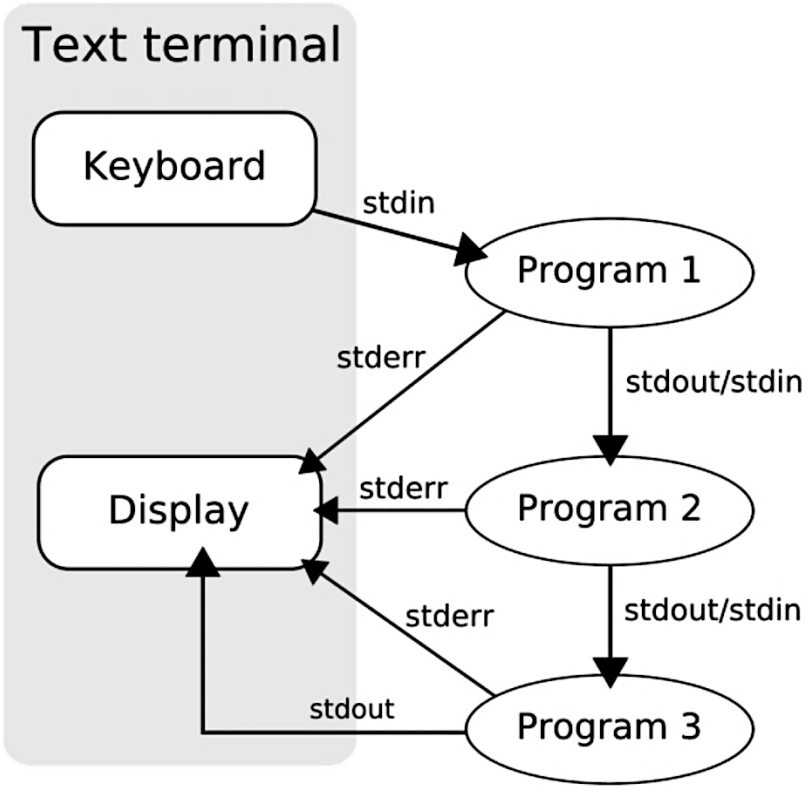


Pipeline

In Unix-like computer operating systems, a pipeline is a sequence of processes chained together by their standard streams, so that the output of each process (stdout) feeds directly as input (stdin) to the next one

The exit status of a pipeline is the exit status of the last process in the pipeline

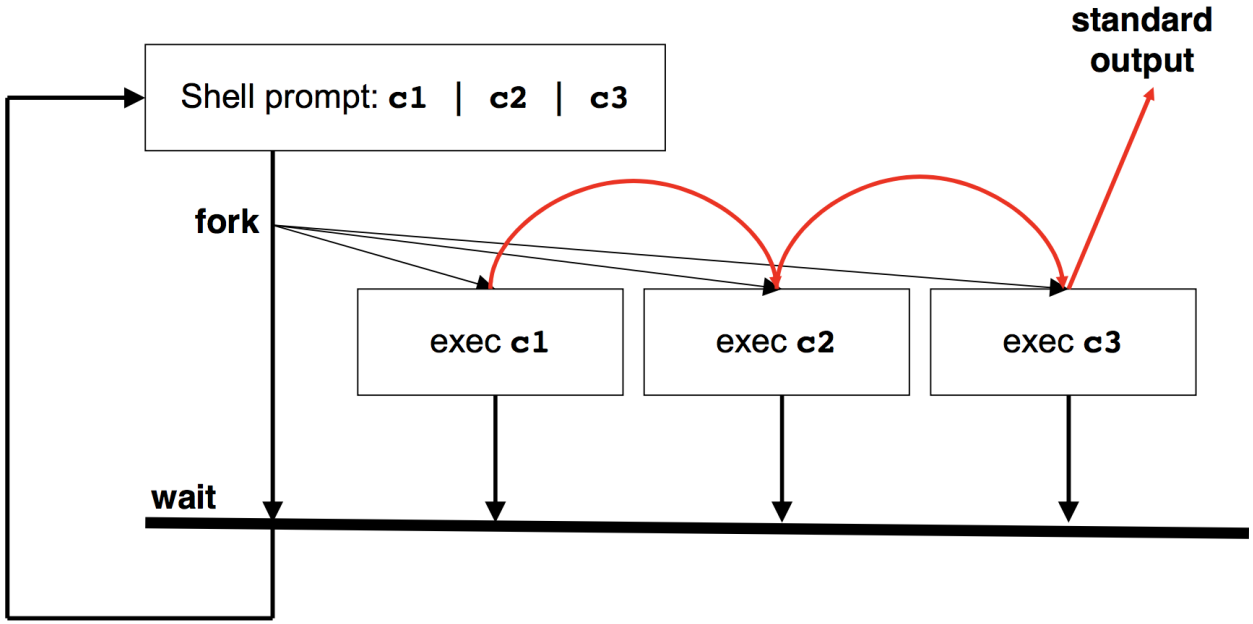
The shell waits for the termination of all the processes in the pipeline before returning the prompt



* https://en.wikipedia.org/wiki/Pipeline_(Unix)

Pipeline execution order

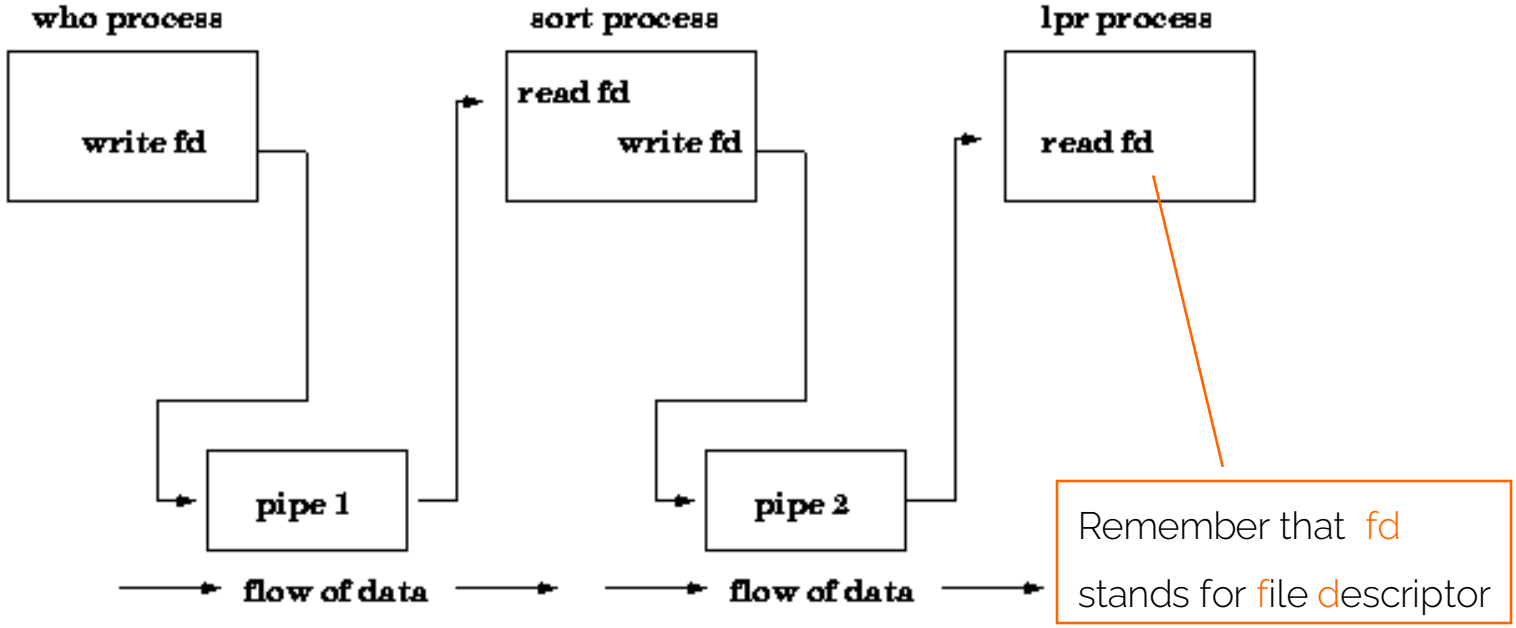
- The order the processes are run does not matter and is not guaranteed
- The shell first creates the pipe, the conduit for the data that will flow between the processes, and then creates the processes with the ends of the pipe connected to them
- For example, the first process that is run may block waiting for input from the second process, or block waiting for the second process to start reading data from the pipe (these waits can be arbitrarily long and do not matter)
- Whichever order the processes are run, the data eventually gets transferred and "everything works"



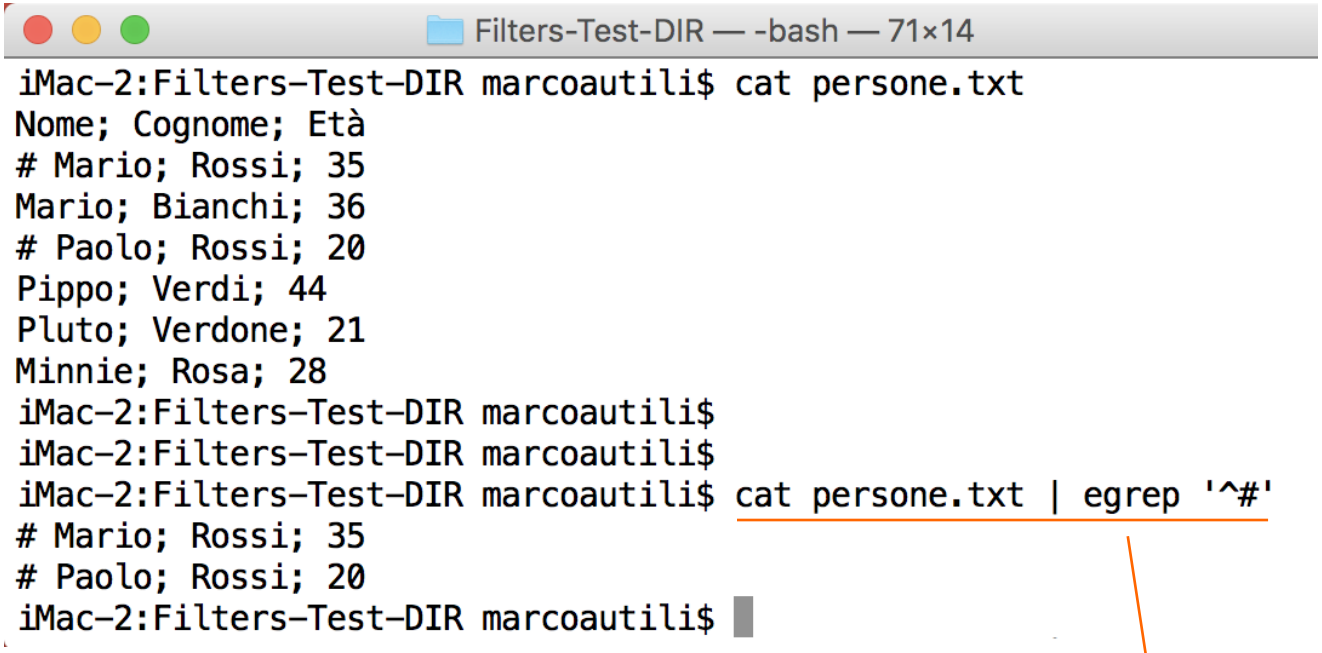
Pipeline

Example: who | sort | lpr

A pipe to send to the default printer (lpr) the ordered (sort) list of all users currently logged on (who)



Pipeline



List only #-commented lines

Pipeline

```
usr — -bash — 68×35
iMac-2:usr marcoautili$ pwd
/usr
iMac-2:usr marcoautili$
iMac-2:usr marcoautili$ ls -la
total 8
drwxr-xr-x@   12 root  wheel    408 Dec 23  2015 .
drwxr-xr-x    32 root  wheel   1156 Sep 20 08:52 ..
drwxr-xr-x     5 root  wheel    170 Aug 23  2015 X11
lrwxr-xr-x     1 root  wheel      3 Oct 18  2015 X11R6 -> X11
drwxr-xr-x     3 root  wheel    102 Sep 20  2015 adic
drwxr-xr-x  1055 root  wheel  35870 Sep  5 13:00 bin
drwxr-xr-x   267 root  wheel   9078 Sep  5 13:01 lib
drwxr-xr-x   186 root  wheel   6324 Sep  5 13:01 libexec
drwxr-xr-x     5 root  wheel    170 Dec 23  2015 local
drwxr-xr-x   243 root  wheel   8262 Sep  5 13:00 sbin
drwxr-xr-x    45 root  wheel   1530 Oct 18  2015 share
drwxr-xr-x     4 root  wheel    136 Oct 18  2015 standalone
iMac-2:usr marcoautili$
iMac-2:usr marcoautili$ ls -la | egrep 'Sep'
drwxr-xr-x    32 root  wheel   1156 Sep 20 08:52 ..
drwxr-xr-x     3 root  wheel    102 Sep 20  2015 adic
drwxr-xr-x  1055 root  wheel  35870 Sep  5 13:00 bin
drwxr-xr-x   267 root  wheel   9078 Sep  5 13:01 lib
drwxr-xr-x   186 root  wheel   6324 Sep  5 13:01 libexec
drwxr-xr-x   243 root  wheel   8262 Sep  5 13:00 sbin
iMac-2:usr marcoautili$
iMac-2:usr marcoautili$ ls -la | egrep 'Sep' | sort
drwxr-xr-x     3 root  wheel    102 Sep 20  2015 adic
drwxr-xr-x    32 root  wheel   1156 Sep 20 08:52 ..
drwxr-xr-x   186 root  wheel   6324 Sep  5 13:01 libexec
drwxr-xr-x   243 root  wheel   8262 Sep  5 13:00 sbin
drwxr-xr-x   267 root  wheel   9078 Sep  5 13:01 lib
drwxr-xr-x  1055 root  wheel  35870 Sep  5 13:00 bin
iMac-2:usr marcoautili$
iMac-2:usr marcoautili$
```

List the directories and files "modified in September"...

Note that it is just an example... it is not always correct. Why?

List the directories and files "modified in September", then sort the output lines

Note that, it sorts alphabetically the whole line... and not by date!

Redirecting VS Pipelining

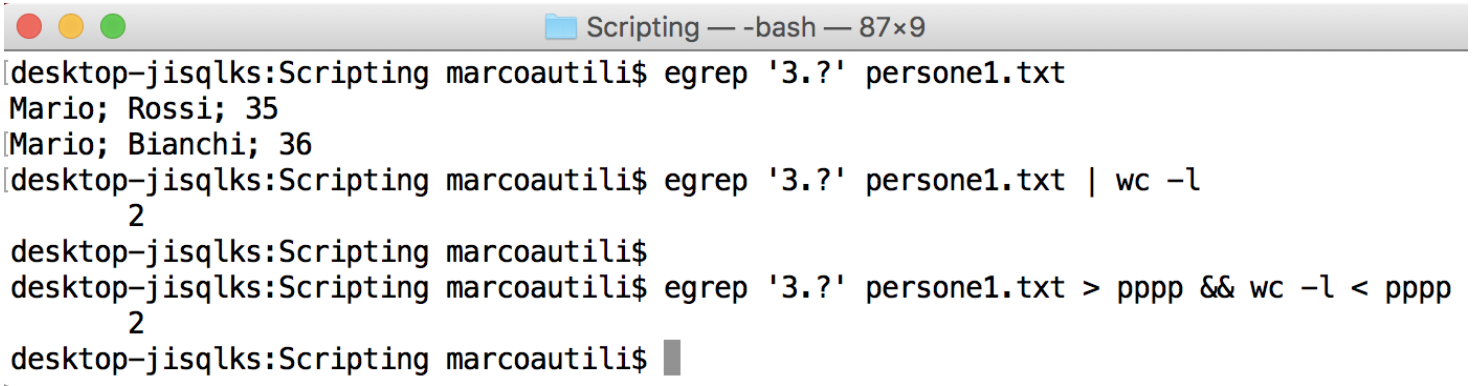
- Roughly speaking, both "do the same basic thing"
  - they redirect the file descriptors processes are connected to; the difference lies in the how
    - A pipe connect the stdout of one process to the stdin of another
    - A redirection redirects from/to a file (> from stdout to a file, < from a file to stdin)

program > file

- You are running program
- The output of program will be placed in file

program1 | program2

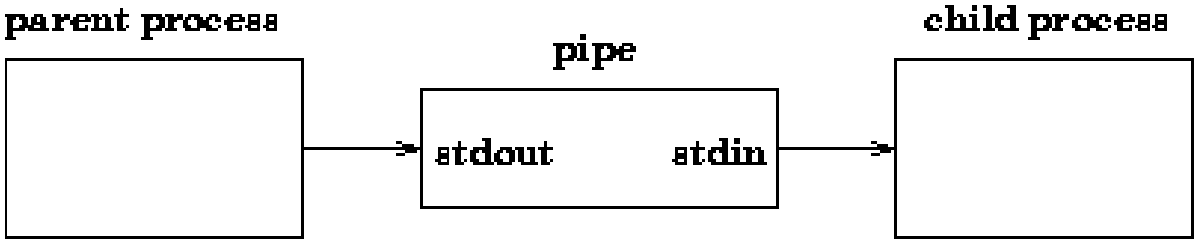
- You may write program1 > temp_file && program2 < temp_file
  - Run program1 and save the output into temp_file
  - Run program2 and pass the contents of temp_file to it



Summing up

When a pipe is used in a Unix-like command line

- the first process is assumed to be writing to stdout
- the second is assumed to be reading from stdin




```
marcoautili — -bash — 92×9
[desktop-jisqlks:~ marcoautili$
[desktop-jisqlks:~ marcoautili$ ps aux | grep Sublime | grep -v grep | awk '{print $2}'
995
[desktop-jisqlks:~ marcoautili$
[desktop-jisqlks:~ marcoautili$
[desktop-jisqlks:~ marcoautili$
[desktop-jisqlks:~ marcoautili$
[desktop-jisqlks:~ marcoautili$ ps aux | grep Sublime | grep -v grep | awk '{print $2}'
 desktop-jisqlks:~ marcoautili$
```

Here, "in between", I closed my text editor Sublime or killed it… more on that later

Note that depending on what version of UNIX-like system you're using, certain programs may behave a little differently (for example, ps might output the PID in column $3)… Again, it is just an example ☺

Activities

- Play with all you have learned so far
- For instance, create files similar to, yet more complex of, persone.txt and combine wildcards, filters, regular expressions, pipelines to perform increasingly complex operations/transformations on it

IMPORTANT

the practical examples I have provided you with so far are NOT enough to pass the exam.

You need to exercise a lot… yourself 

This applies to the coming arguments as well 

Programs and Processes (from PART I)

Program

- an executable file residing on disk in a directory
    - A program is read into memory and is executed by the kernel as a result of one of the seven exec functions

Processes and Process ID

- An executing instance of a program is called a process,
    - Sometimes the term task is used to refer to a program that is being executed ☹
- The UNIX System guarantees that every process has a unique numeric identifier called the process ID
    - the process ID is always a non-negative integer

Process Control

- There are three primary functions for process control: fork, exec, and waitpid
- The exec function has seven variants, but we often refer to them collectively as simply the exec function

Multitasking operating systems

- Many processes or tasks might be running “at the same time”
- At the same time does not necessarily mean that processes are executing at exactly the same time
  - it does mean that more than one processes can be part-way through execution at the same time, and that more than one process is advancing over a given period of time
  - new processes “can” interrupt already started ones before they finish, instead of waiting for them to end. As a result, a computer executes segments of multiple processes in an interleaved manner, while the processes share common processing resources such as central processing units (CPUs) and main memory
- Additional terminologies should be mentioned (and compared) here:
  - simultaneously, concurrently, parallel execution
- From the command line, the top program displays and updates sorted information about processes directly on the terminal (press q to exit)

Processes

ps [aux] (stand for process status)

- show only the processes running in your current terminal (usually, should not be too many)
- passing the argument aux a complete system view is shown

marcoautili — -bash — 155×34

```
iMac-2:~ marcoautili$
iMac-2:~ marcoautili$
iMac-2:~ marcoautili$ ps
  PID TTY           TIME CMD
  320 ttys000    0:00.04 -bash
iMac-2:~ marcoautili$
iMac-2:~ marcoautili$
iMac-2:~ marcoautili$
iMac-2:~ marcoautili$ ps aux
USER               PID  %CPU %MEM      VSZ    RSS   TT  STAT STARTED      TIME COMMAND
marcoautili        255   4.1  0.5  2958444 182976   ??  R     1:17PM   0:14.54 /Applications/Utilities/Terminal.app/Contents/MacOS/Terminal -psn_0_40970
marcoautili        584   0.2  0.2  3781628  82448   ??  S     1:26PM   0:11.85 /System/Library/CoreServices/Dock.app/Contents/Resources/DashboardClient.app/
Contents/MacOS/DashboardClient
_windowserver      184   0.2  0.5  6670308 163776   ??  Ss    1:17PM   1:31.71 /System/Library/Frameworks/ApplicationServices.framework/Frameworks/CoreGraph
ics.framework/Resources/WindowServer -daemon
marcoautili        387   0.1  0.8  4352980 266204   ??  S     1:18PM   3:53.22 /Applications/Dropbox.app/Contents/MacOS/Dropbox
marcoautili        771   0.0  0.1  2611364  31860   ??  S     2:18PM   0:00.46 /Applications/Microsoft PowerPoint.app/Contents/SharedSupport/Office365Servic
eV2.app/Contents/MacOS/Office365ServiceV2
root               766   0.0  0.0  2508128   4464   ??  Ss    2:17PM   0:00.01 /usr/sbin/ocspd
marcoautili        761   0.0  0.1  3629928  20912   ??  Ss    2:16PM   0:00.09 /System/Library/Frameworks/WebKit.framework/Versions/A/XPCServices/com.apple.
WebKit.WebContent.xpc/Contents/MacOS/com.app
marcoautili        717   0.0  0.4  4134396 135988   ??  Ss    2:08PM   0:01.12 /System/Library/StagedFrameworks/Safari/WebKit.framework/Versions/A/XPCServic
es/com.apple.WebKit.WebContent.xpc/Contents/
marcoautili        708   0.0  0.0  2507884   4912   ??  Ss    2:06PM   0:00.02 /System/Library/Frameworks/AudioToolbox.framework/XPCServices/com.apple.audio
.ComponentHelper.xpc/Contents/MacOS/com.appl
marcoautili        707   0.0  0.0  2472520   6356   ??  Ss    2:06PM   0:00.01 /System/Library/Frameworks/AudioToolbox.framework/XPCServices/com.apple.audio
.SandboxHelper.xpc/Contents/MacOS/com.apple.
marcoautili        706   0.0  0.0  3646884  15020   ??  Ss    2:06PM   0:00.03 /System/Library/StagedFrameworks/Safari/WebKit.framework/Versions/A/XPCServic
es/com.apple.WebKit.Databases.xpc/Contents/M
marcoautili        690   0.0  0.0  2542084   9380   ??  S     1:54PM   0:00.08 /System/Library/PrivateFrameworks/AOSKit.framework/Versions/A/XPCServices/com
.apple.iCloudHelper.xpc/Contents/MacOS/com.a
marcoautili        666   0.0  0.0  2520472  13620   ??  S     1:44PM   0:00.03 /System/Library/Frameworks/CoreServices.framework/Frameworks/Metadata.framewo
rk/Versions/A/Support/mdworker -s mdworker -
marcoautili        619   0.0  0.1  2614612  30920   ??  Ss    1:31PM   0:01.37 /System/Library/Frameworks/Foundation.framework/Versions/C/XPCServices/Lookup
```

Kill command

kill [signal] <PIDs>

- The **kill** utility sends the signal [signal] to the processes specified by the <PIDs> operands
- A user can kill her own processes
- Only the super-user may send signals to other users' processes

- The PID -1 have special meanings:
  - if superuser, broadcast the signal to all processes
  - otherwise, broadcast to all processes belonging to the user

EXIT STATUS

- The **kill** utility exits 0 on success, and > 0 if an error occurs
- Some shells may provide a builtin **kill** command which is similar or identical to this utility (see previous slides on builtins)

Killing a process

```
marcoautili — -bash — 141×10
iMac-2:~ marcoautili$
iMac-2:~ marcoautili$ ps aux | egrep 'Sublime'
marcoautili      1102   0.0  0.0  2472728    880 s000  S+    6:06PM   0:00.00 egrep Sublime
marcoautili      1092   0.0  0.5  2900864 162476   ??  S     6:05PM   0:02.40 /Applications/Sublime Text 2.app/Contents/MacOS/Sublime Text 2
iMac-2:~ marcoautili$
iMac-2:~ marcoautili$ kill 1092
iMac-2:~ marcoautili$
iMac-2:~ marcoautili$ ps aux | egrep 'Sublime'
marcoautili      1121   0.0  0.0  2447128    848 s000  S+    6:08PM   0:00.00 egrep Sublime
iMac-2:~ marcoautili$
```

If not specified, the signal "15" (SIGTERM) is send by default to the process,

i.e., it is equivalent to

kill -15 1092

kill -TERM 1092

kill –s TERM 1092

- It can happen that, by simply sending the signal "-15", the process does not terminate
- In this case, sending the signal "-9" might be of help: it forces the termination
- Pay attention when killing processes… You should know what you are doing ☺

Some of the more commonly used signals

1	HUP (hang up) (see https://en.wikipedia.org/wiki/SIGHUP)
2	INT (interrupt)
3	QUIT (quit)
6	ABRT (abort)
9	KILL (non-catchable, non-ignorable kill)
14	ALRM (alarm clock)
15	TERM (software termination signal)

Killing a process

Killing multiple processes

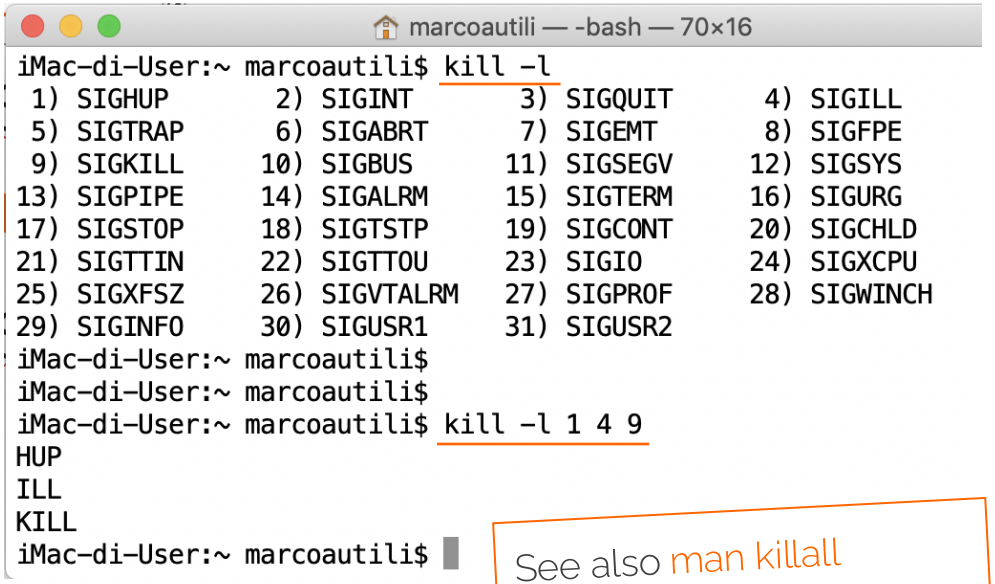
- kill -9 <PID1> <PID2> … <PIDn>

Getting the name of a signal

- kill -l <signal number>

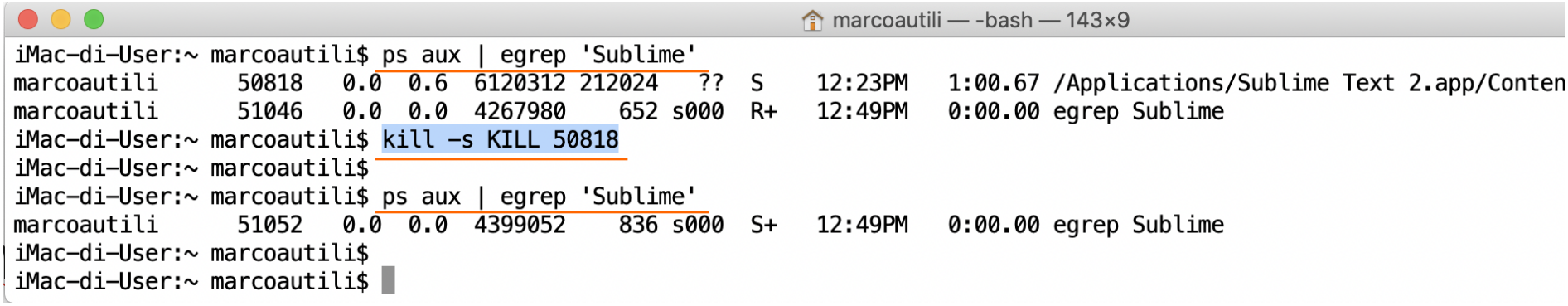
Sending signal by name

- kill -s <signal name> <PID>

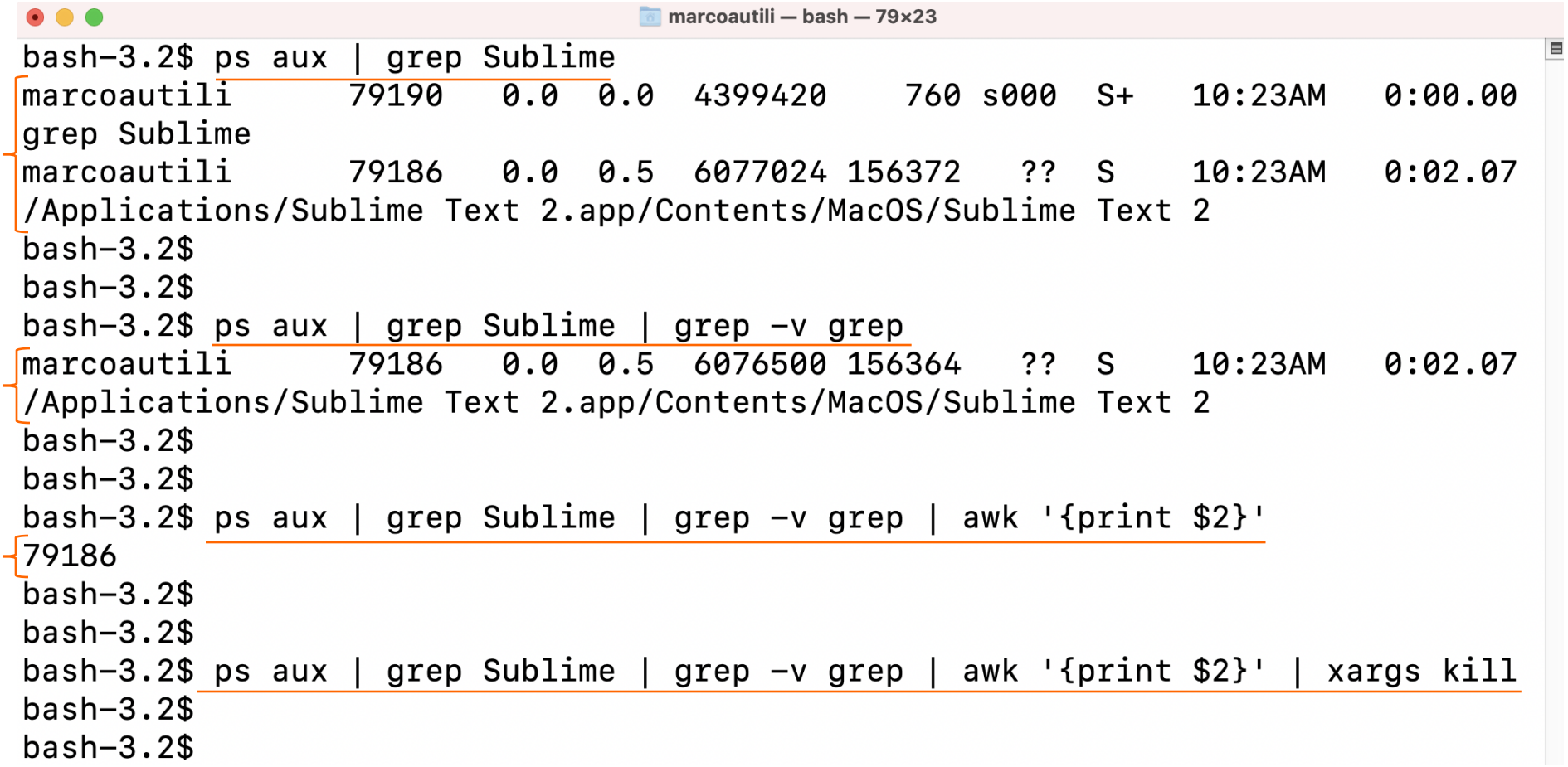


See also man killall

(kill processes by name)



Back to the pipelining example



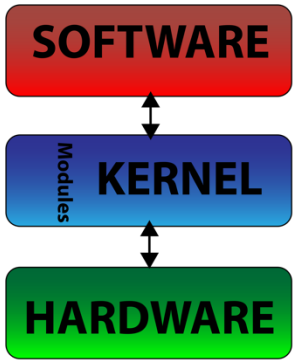
Signals (more on)

- Signals are a limited form of inter-process communication (IPC), typically used in Unix, Unix-like, and other POSIX-compliant operating systems

- A signal is an asynchronous notification sent to a process (or to a specific thread within the same process) in order to notify it of an event that occurred

- When a signal is sent, the operating system interrupts the target process' normal flow of execution to deliver the signal. Execution can be interrupted during any non-atomic instruction
  - If the process has previously registered a signal handler, that routine is executed. Otherwise, the default signal handler is executed

- Signals are similar to interrupts, the difference being that
  - interrupts are mediated by the CPU and handled by the kernel
  - signals are mediated by the kernel (possibly via system calls) and handled by processes



https://en.wikipedia.org/wiki/Signal_(IPC)
https://en.wikipedia.org/wiki/Interrupt

Signals (more on)

- The kernel may pass an "interrupt" as a "signal" to the process that caused it (typical examples are SIGSEGV, SIGBUS, SIGILL and SIGFPE)

- A process's execution may result in the generation of a hardware exception, for instance, if the process attempts to divide by zero or incurs a page fault
  - In fact, if a process attempts to divide an integer by zero, a Floating-Point Exception is generated, and the kernel sends the SIGFPE signal to the process
  - Also, if process makes an invalid virtual memory reference, or segmentation fault, i.e., it performs a SEGmentation Violation, the kernel sends the SIGSEGV signal to the process

- The exact mapping between signal names and exceptions is obviously dependent upon the CPU, since exception types differ between architectures

https://en.wikipedia.org/wiki/Signal_(IPC)

Additional material

- Tutorial: Master The Command Line

  https://ryanstutorials.net/linuxtutorial/

- Turorial: The Unix Shell

  https://swcarpentry.github.io/shell-novice/

- Command Line Cheatsheet

  http://cht.sh

- Explain Shell

  https://explainshell.com

References

- Reference textbook, [APUE3ed] Advanced Programming in the UNIX Environment, 3rd Ed. Richard Stevens, Steven Rago

- http://ryanstutorials.net/linuxtutorial/

- http://faculty.cs.niu.edu/~freedman/

- https://www.scribd.com/document/387330980/amministrare-gnu-linux-v4-0-web-cover-bis-pdf