

CORSO DI METODI MOLECOLARI E BIOINFORMATICA

LM Biologia Evoluzionistica, Università di Padova

Dr. Enrico Gaffo, Dr. Silvia Orsi,

Prof. Stefania Bortoluzzi

Esercitazione 3 “Unix/Linux e Bash”

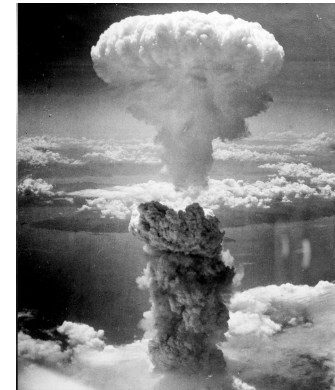
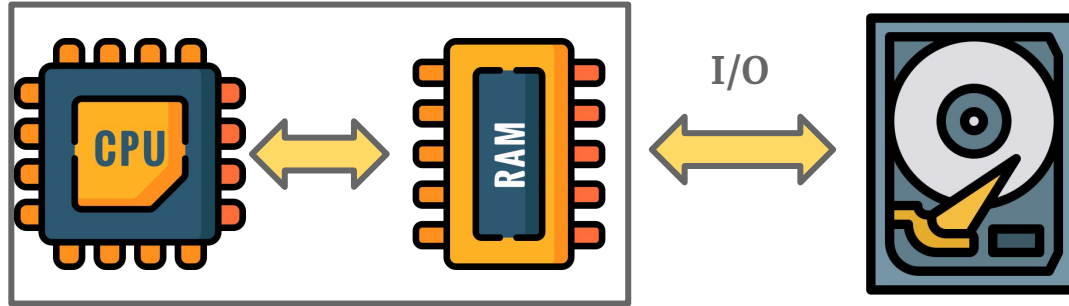
Padova, 9 Novembre 2023

Obiettivo dell'esercitazione

- Familiarizzare con l'interfaccia da linea di comando dei sistemi Linux (e Unix-like)

Basic schema of a computer machine:

the Von Neumann architecture



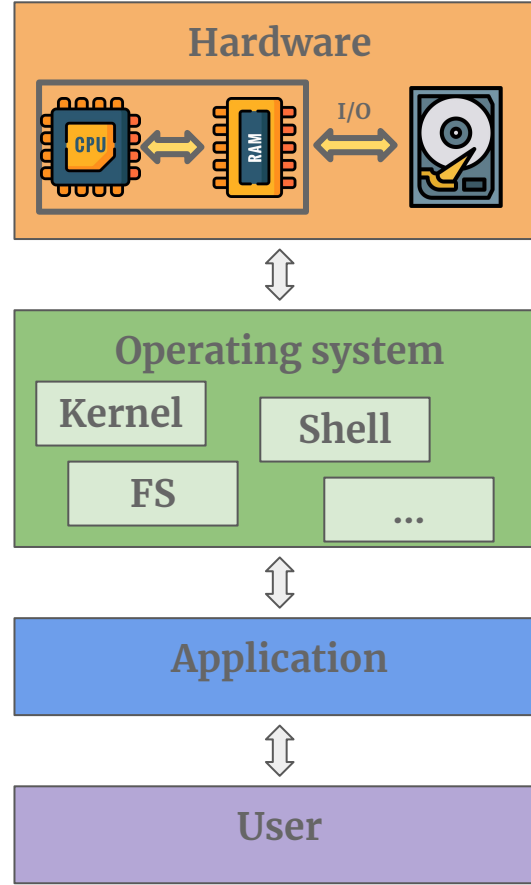
Operating system (OS)



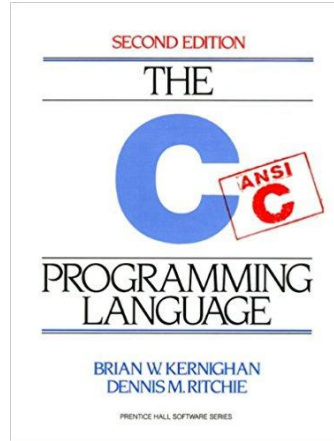
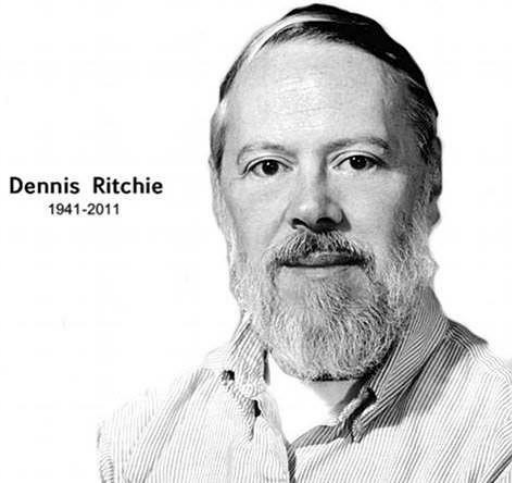
chromeOS



ubuntu.



Dennis Ritchie

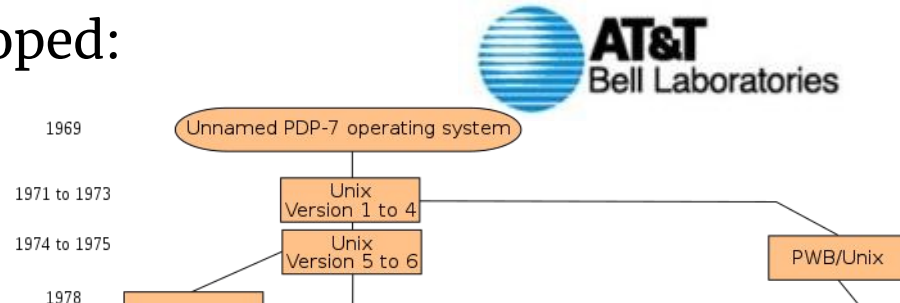


Ken Thompson



In the early '70s invented and developed:

- the C programming language
- the Unix OS



The Unix philosophy

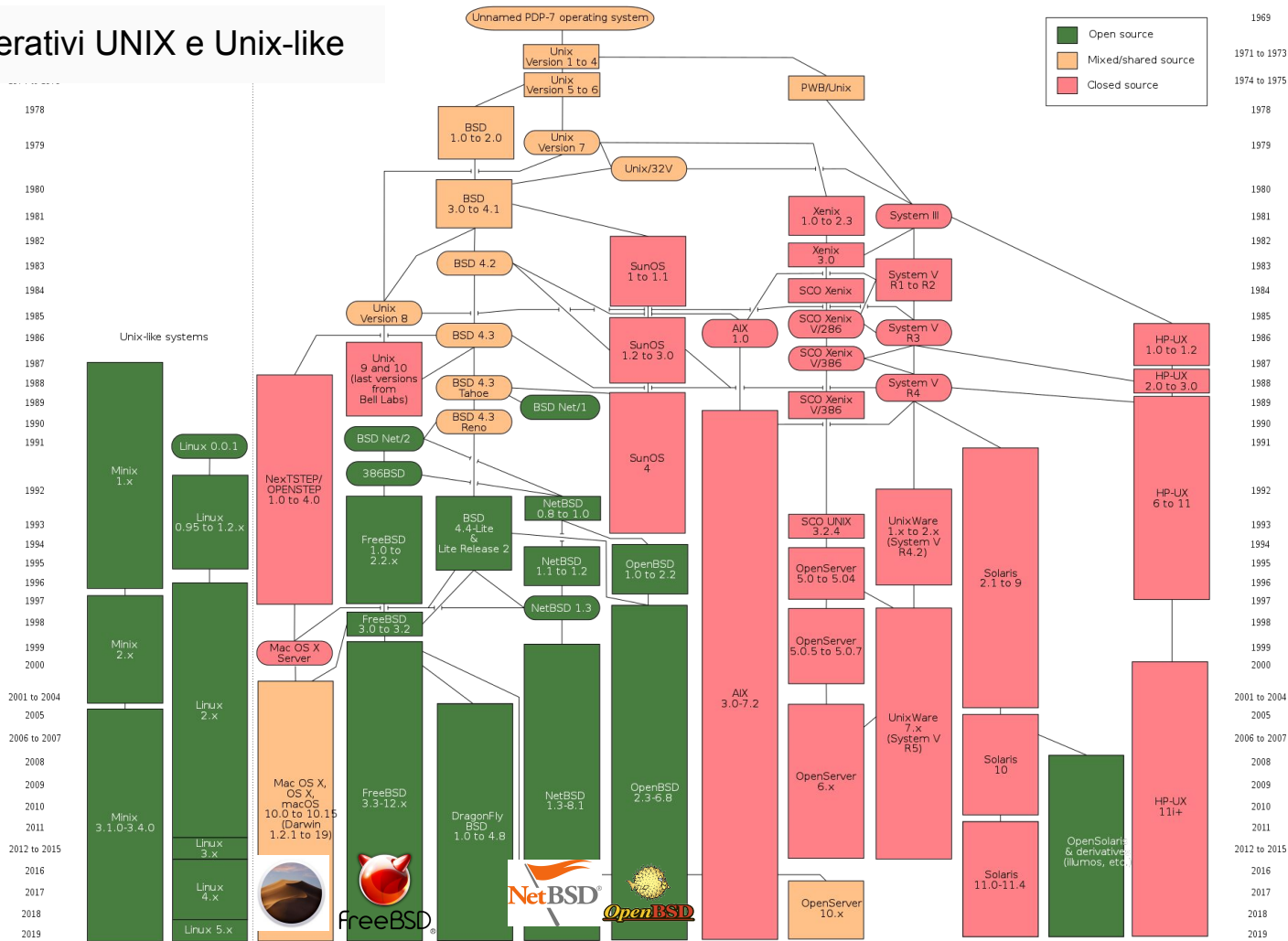
Building simple, short, clear, modular, and extensible code that can be easily maintained and repurposed by developers other than its creators.

Favor composability

i.e. software can be assembled in various combinations to satisfy specific
user requirements

(we will exploit that, f.i. when using the “pipe” operator)

Sistemi operativi UNIX e Unix-like



The GNU Project

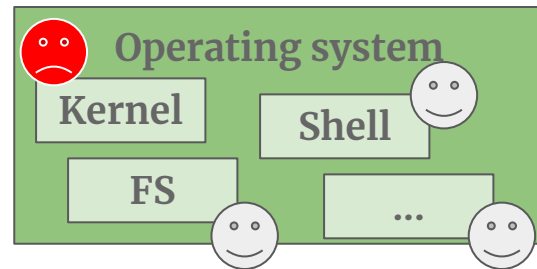
GNU = "GNU's Not Unix!"

Started in 1983, had the goal of creating a "complete Unix-compatible software system" composed entirely of free software



Richard Stallman

By the early 1990s, many of the programs required in an operating system (such as libraries, compilers, text editors, a command-line shell, and a windowing system) were completed...but the kernel was incomplete !



Linus Torvalds



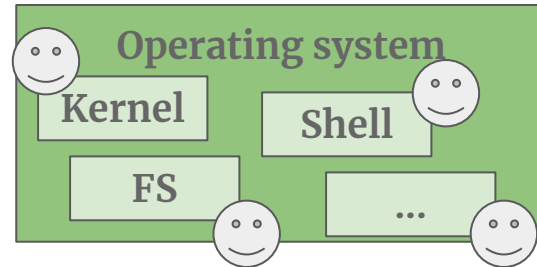
GNU/Linux OS

In 1991, wrote its own operating
system kernel “Unix-like”
(motivated by licensing issues)

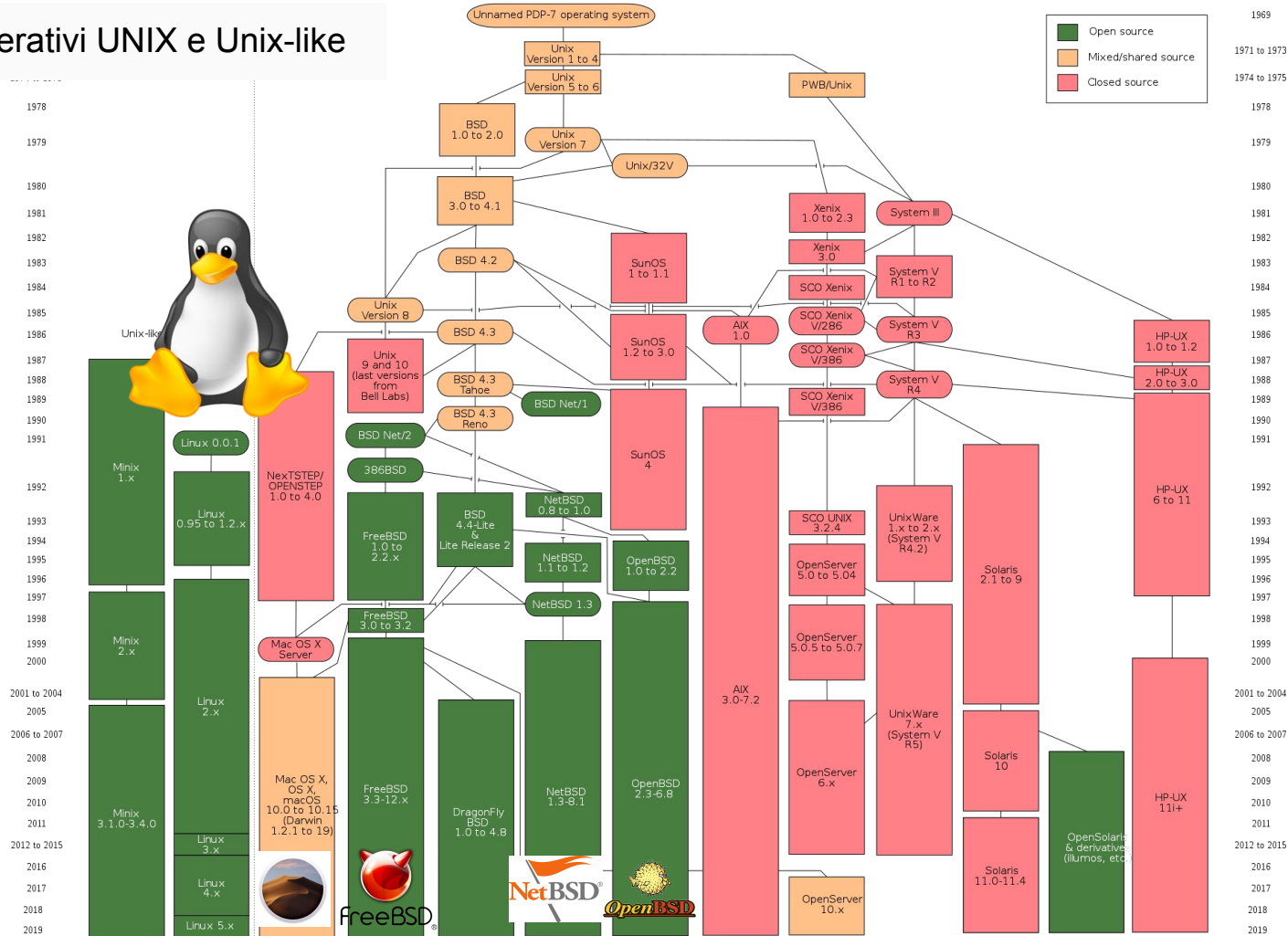
Linux (kernel)



GNU OS components



Sistemi operativi UNIX e Unix-like



Distribuzioni linux



debian



ubuntu



redhat



fedora



slackware
linux



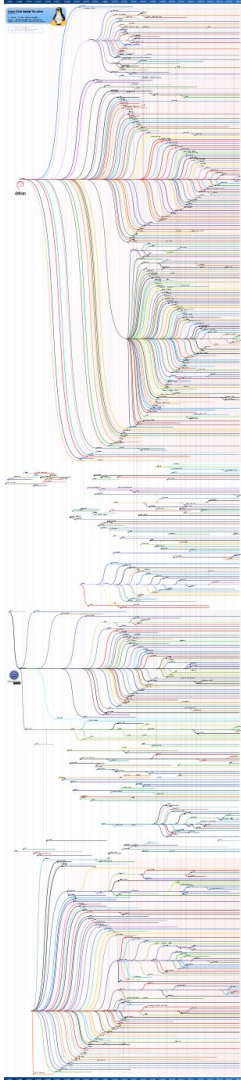
archlinux

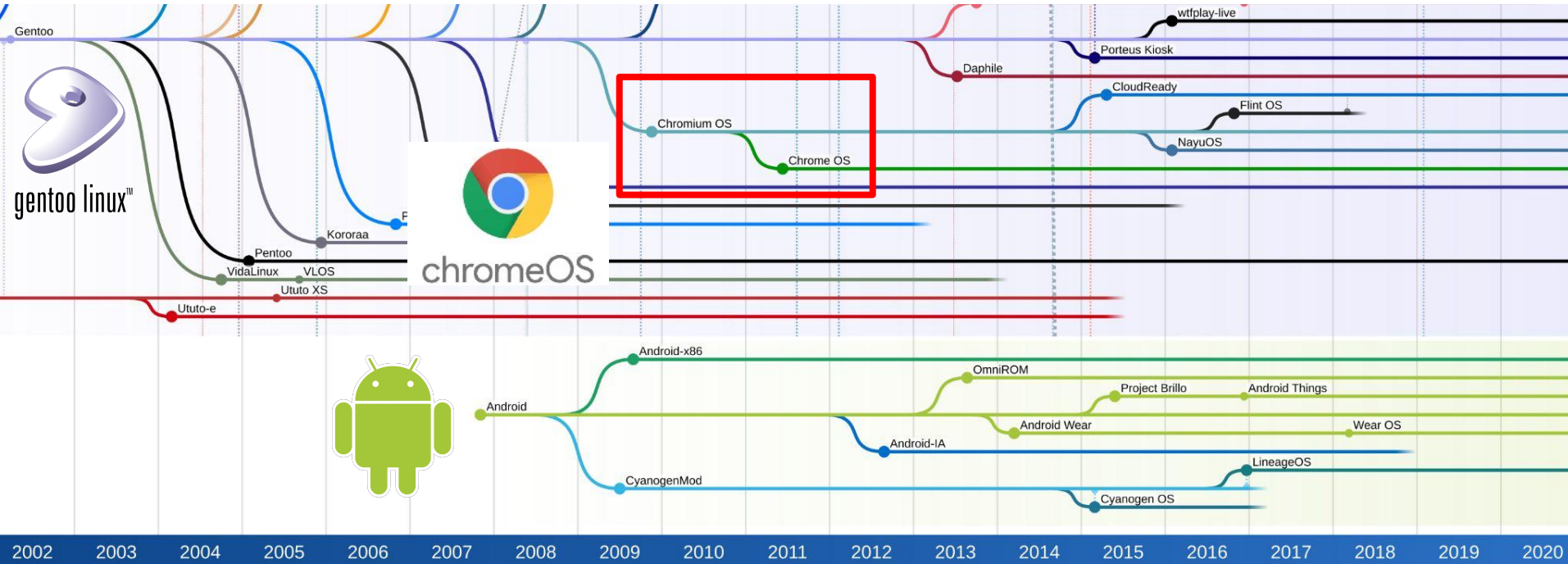


gentoo linux™



https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg





Perchè usare Linux per la bionformatica?

- La maggioranza dei tool bioinformatici sono sviluppati per Linux
- È open source e (spesso) free
- “Compatibile” tra sistemi desktop e server
- Command line (Shell) con molti strumenti per lavorare su file di testo
- Automatizzare software di analisi (composability!)

Shell: a user interface

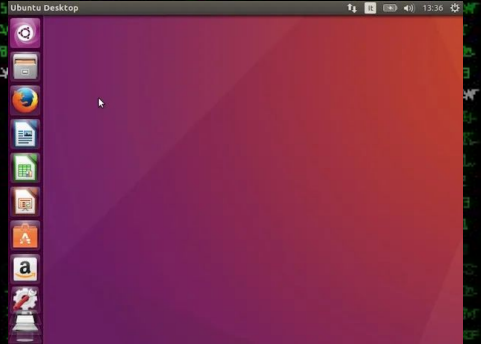
The outermost layer around the operating system

Allows the user to interact with the OS

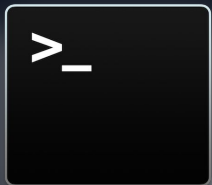
>_

Command-line shells

Graphical shells



Command line interface



```
< CAN YOU MAKE ME A SANDWICH?>
```

```
< NO >
```

```
< SUDO MAKE ME A SANDWICH >
```

```
< OK >
```



What is the command line?

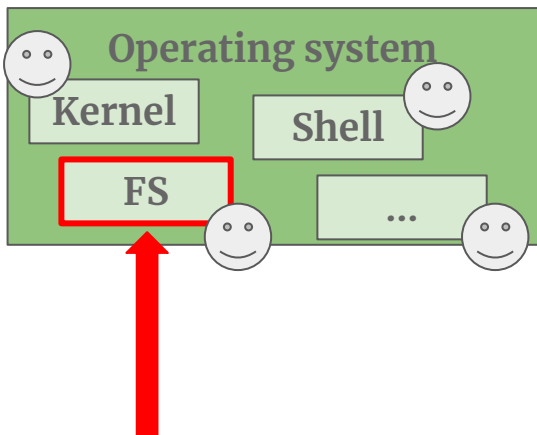
The window, which is usually called the command line or command-line interface, is a text-based application for viewing, handling, and manipulating files on your computer. It's much like Windows Explorer or Finder on the Mac, but without the graphical interface. Other names for the command line are:

- *Cmd*
- *CLI*
- *Prompt*
- *Console*
- *terminal*

Perchè usare la CLI Linux?

- Alte performance computazionali
- Richiede meno risorse computazionali (no GUI)
- I comandi non cambiano nonostante gli aggiornamenti
- Automatizzare i passaggi di un'analisi (comporre pipeline)
- Flessibilità
- Facile lavorare su macchine in remoto

Wander around ...



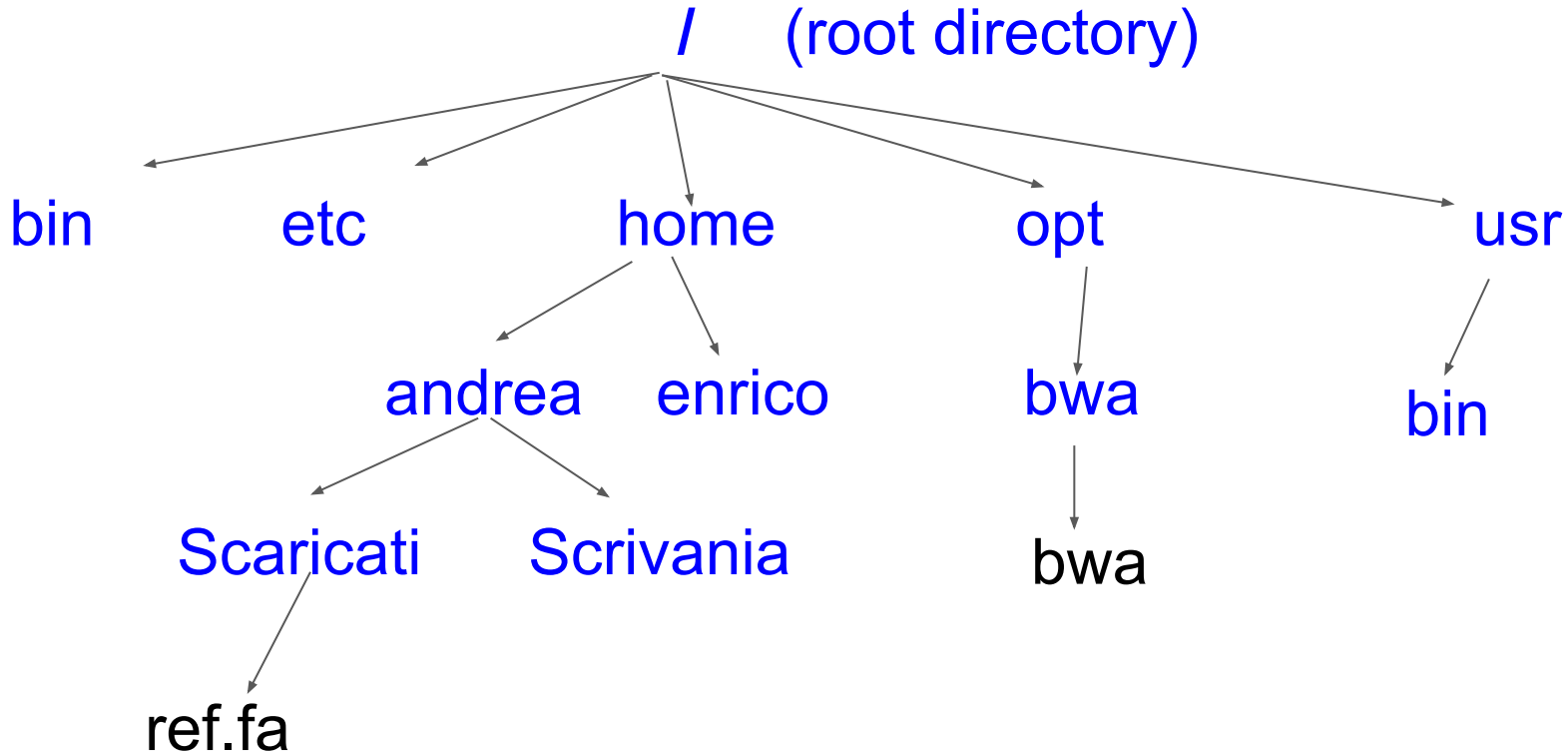
... a hierarchical file system ...

```

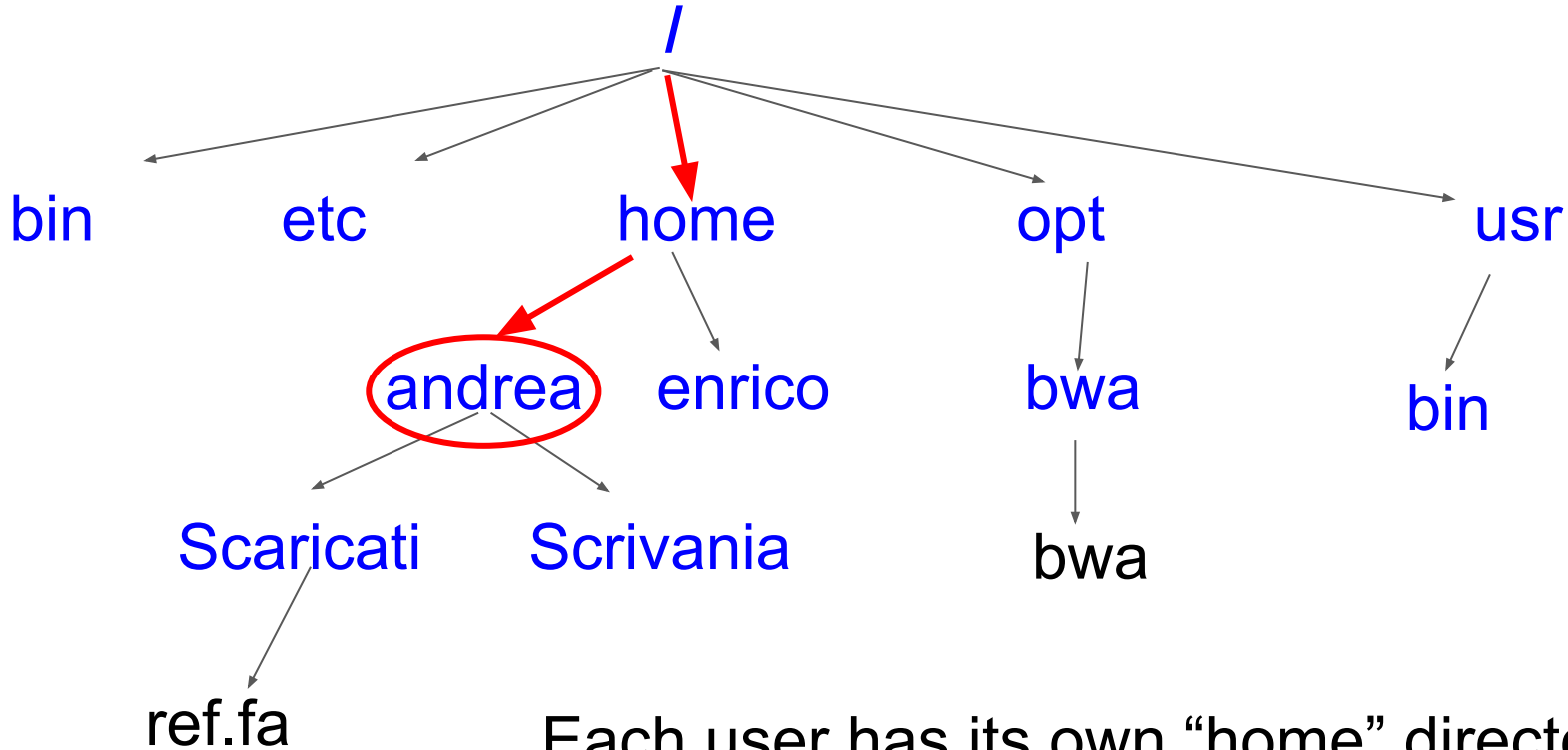
      .-""-.
     /   \  " _ , --- "" = -- , _ "   \   /
    /       \ : - :           : - :   \   /
   /         \|o\ : :         \|o\   \   /
  /           := "~\           ~"=:   \   /
 /             (\               )\     /
/              |              | \     /
.-""-.        /              \  /     \  .-""-.
{            }--| / , .- - . , \ |--{            }-----
)      ( _ ) _ ) \_ / `~---==~-~\ \_/ ( _ ( _ ) (
( ... using the command line ! wof wof ! )
)
'-----'
```

Hierarchical file system

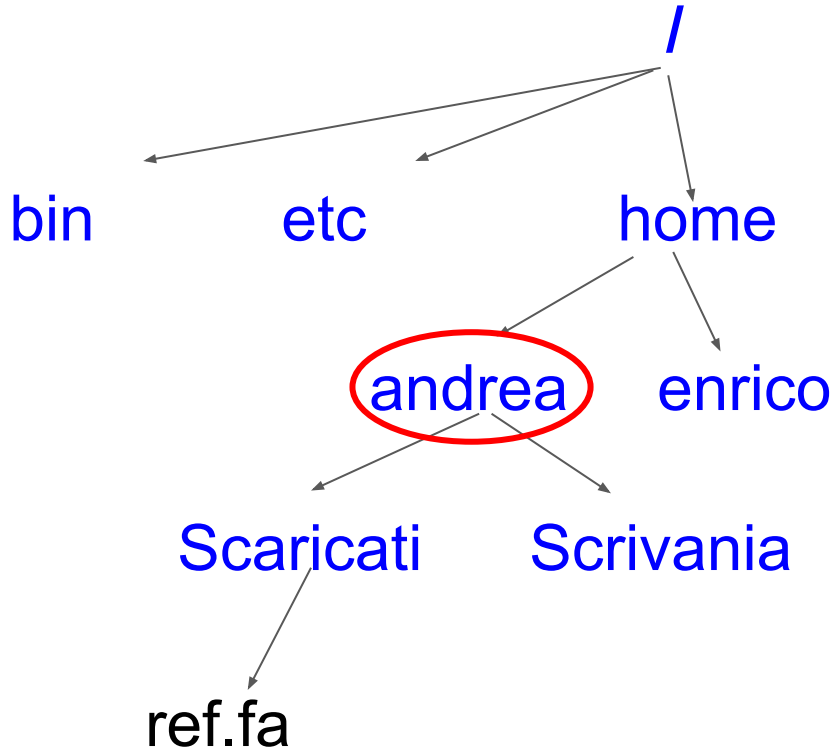
https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard



The “home” Directory

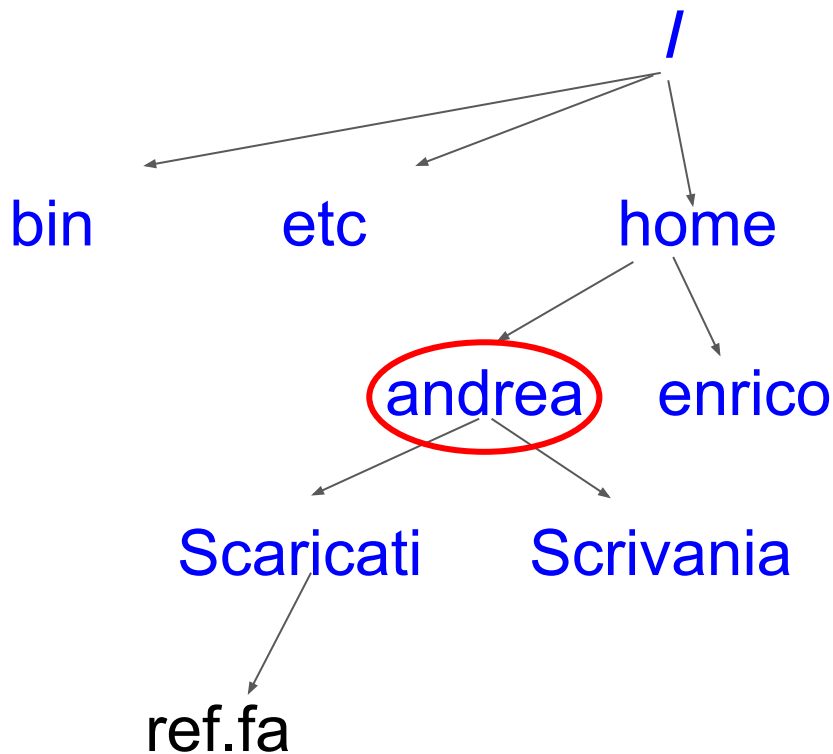


The current/working directory



- The shell is always “positioned” into a **current** directory.
- When logging into the system, you are positioned into your user’s *home*

The current/working direcotry



Print working directory:

\$pwd

/home/andrea

List files in current directory:

\$ls

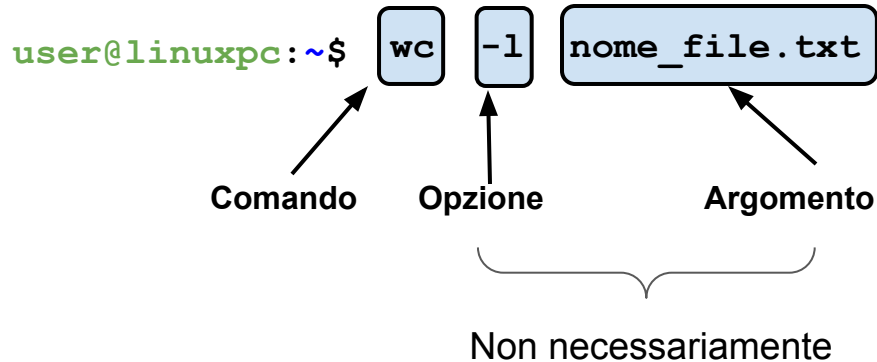
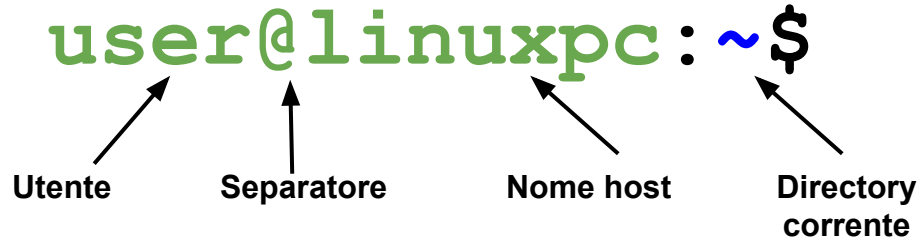
Scaricati Scrivania

Change working directory

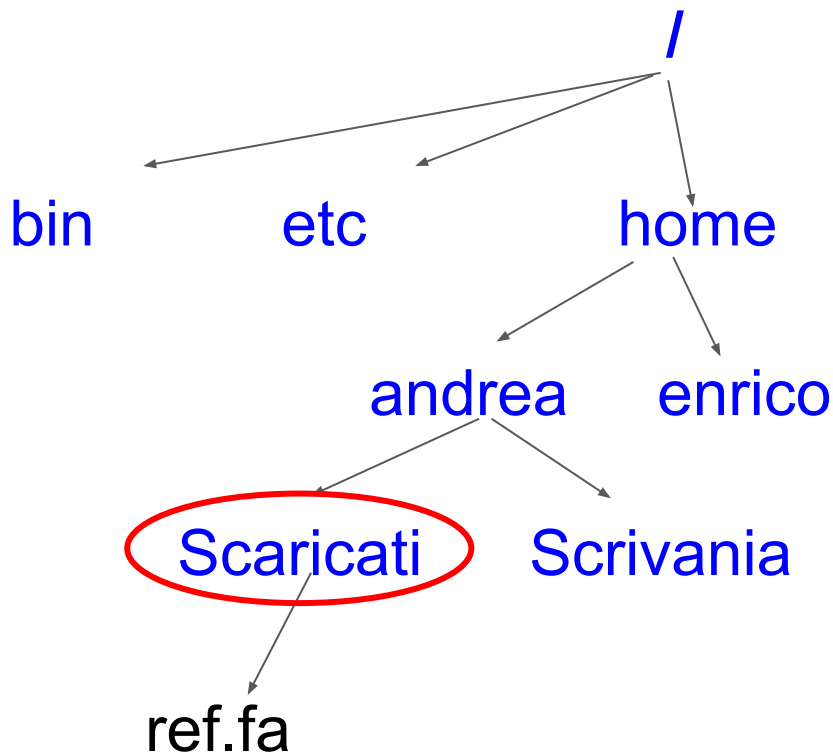
\$cd Scaricati

Anatomia di un comando

Prompt dei comandi

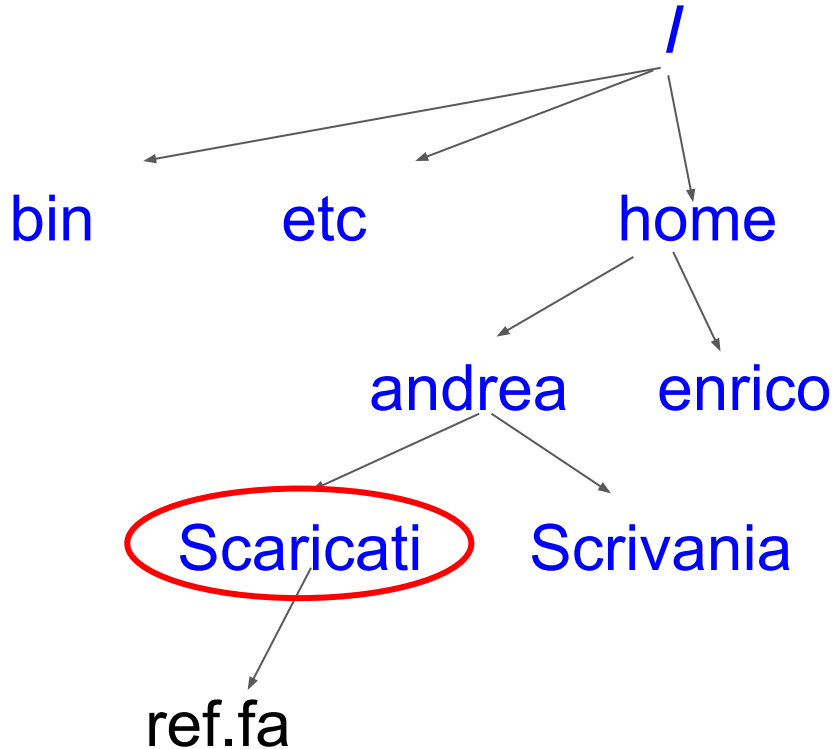


The current/working direcotry



```
$pwd  
/home/andrea/Scaricati  
$ls  
ref.fa
```

Absolute and Relative Paths



An **absolute path** starts from the **root** directory (/)

/home/andrea

/opt/bwa/bwa

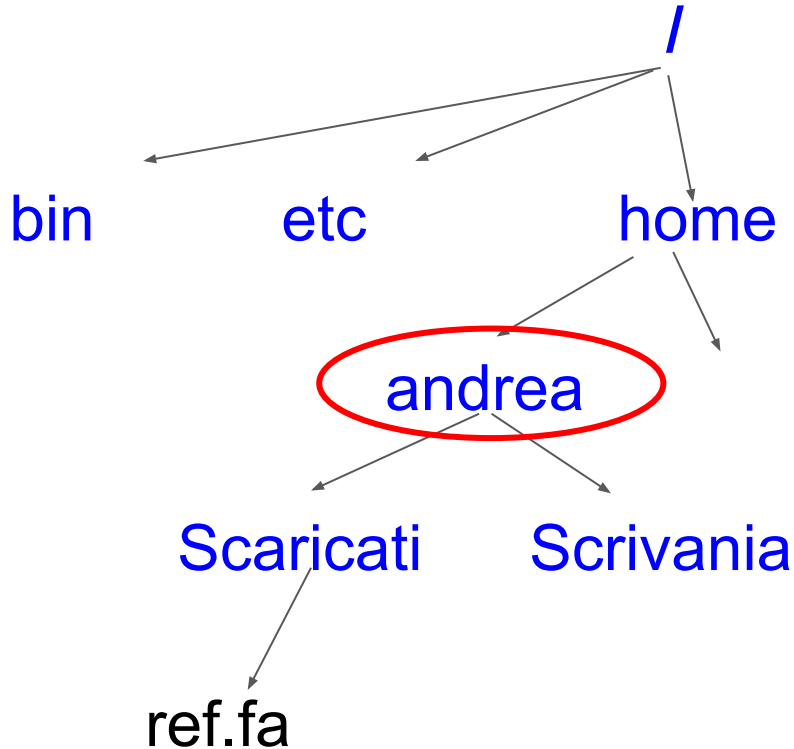
/home/andrea/Scaricati/ref.fa

A **relative path** start from the **current** directory

Scaricati/ref.fa

Relative paths depend on the current directory

Absolute and Relative Paths

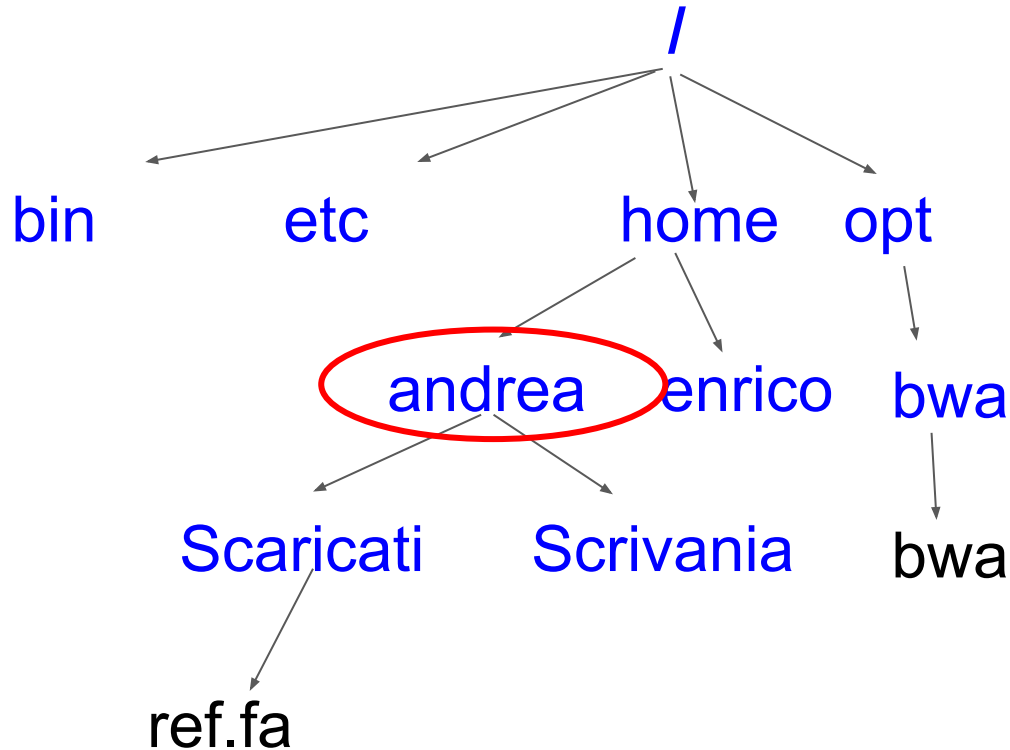


Relative paths start from the current directory:

```
$cd /home/andrea
```

```
$less Scaricati/ref.fa
```

Relative Paths and special directories “.” and “..”



“.” and “..” are always present
.. is the parent directory
. is the current directory

```
$pwd
/home/andrea
$ls ..
andrea enrico
$cd ..
$pwd
/home/
$ls ../opt/bwa
bwa
$cd
$pwd
/home/andrea
```

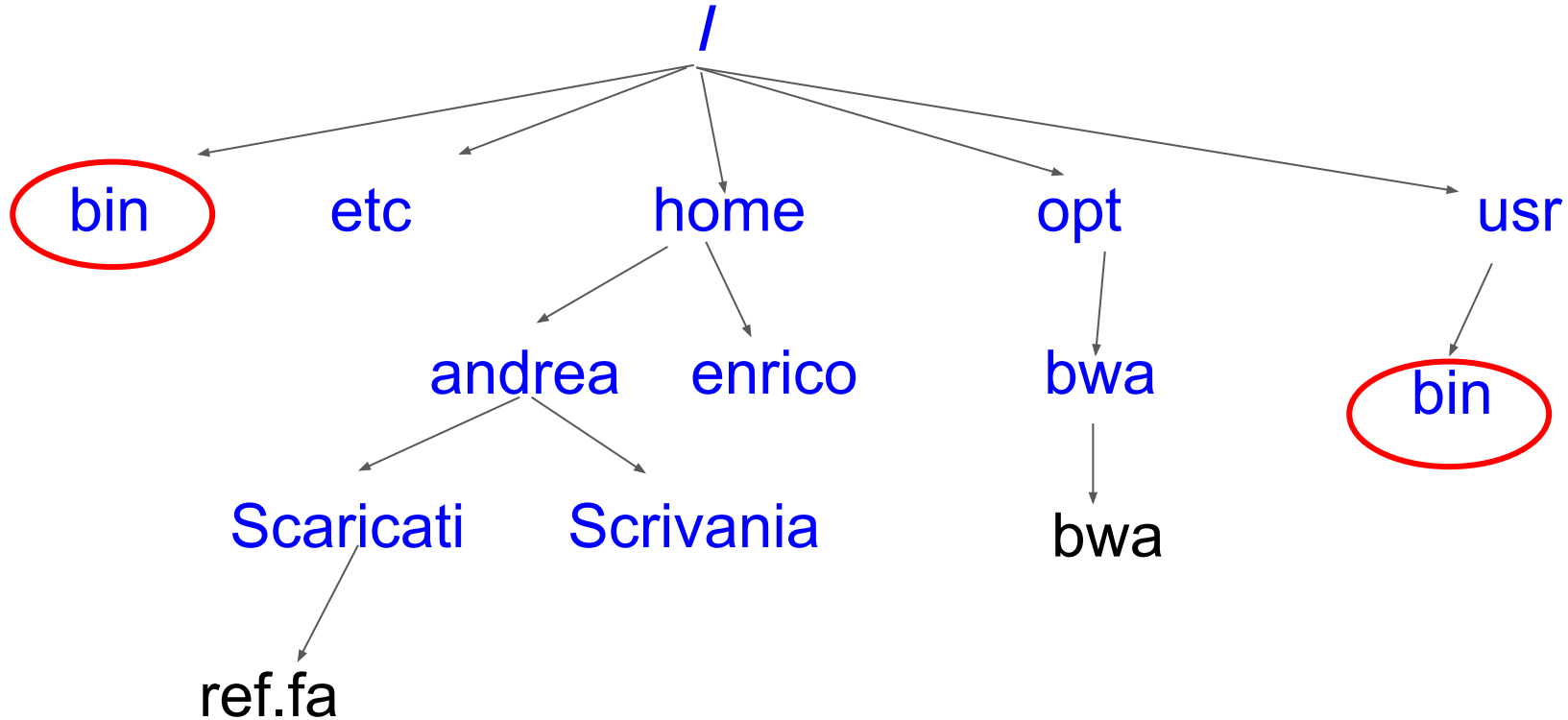
The PATH environment variable

PATH is an **environment variable** in Linux and other Unix-like operating systems that tells the shell which directories to search for executable files (i.e., ready-to-run programs) in response to commands issued by a user.

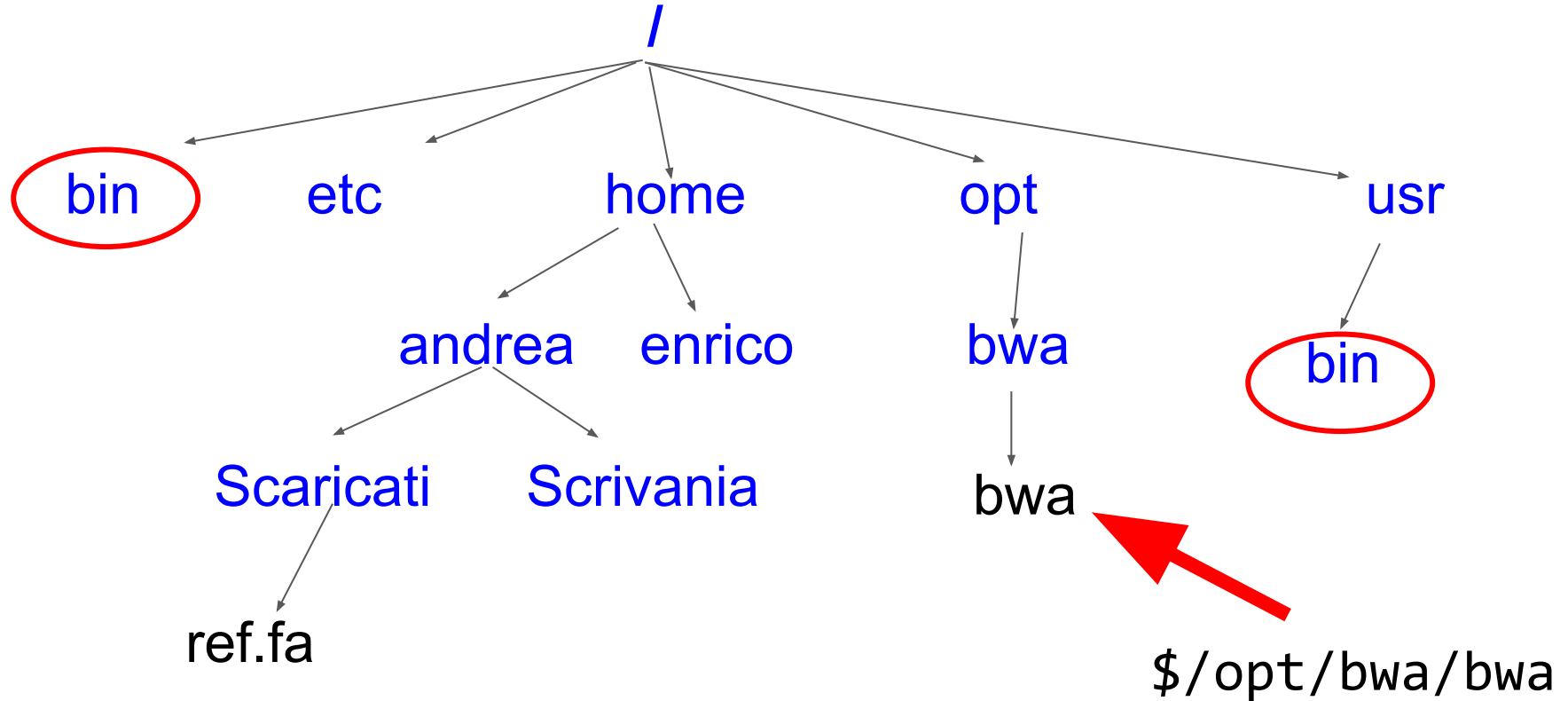
```
user@linuxpc: ~/$ echo $PATH
```

Binaries are being searched for from left to right

The PATH environmental variable



How to execute a program/command not in the PATH?



Preparazione dati per esercitazione

Scaricare nella propria home directory il file dell'esercitazione 3 dal sito http://compgen.bio.unipd.it/~stefania/Didattica/AA2023-2024/MMOL_BIOINFO_BE/esercitazione3.zip

- Aprire un terminale: di default si è posizionati nella propria home directory:

N.B.: il comando `cd` senza argomenti ci porta alla home directory dell'utente

Accertarsi che il file sia presente nella directory corrente e/o **di essere nella directory giusta** (suggerimento: `ls`, `pwd`)

- Decomprimere l'archivio contenente i file per l'esercitazione usando il comando `unzip`:

```
$ unzip esercitazione3.zip
```

- Entrare nella directory `esercitazione4` con il comando `cd` e il path relativo verso la cartella `esercitazione3`:

```
$ cd esercitazione3
```


Preparazione dati per esercitazione

- Vedere il contenuto della directory. Usiamo l'opzione “-l” del comando **ls** per avere un file per riga:

```
$ ls -l
```

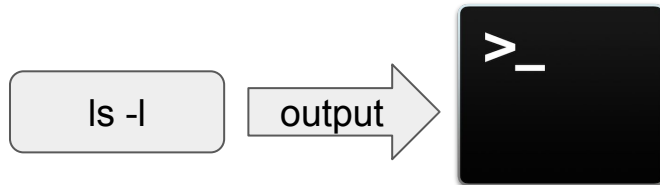
Verranno visualizzati **sul terminale** i nomi dei file presenti nella directory, tra cui 10 file in formato FASTA contenenti sequenze di trascritti riconducibili a geni coinvolti nel cancro al seno.

Preparazione dati per esercitazione

- Vedere il contenuto della directory. Usiamo l'opzione “-l” del comando **ls** per avere un file per riga:

```
$ ls -l
```

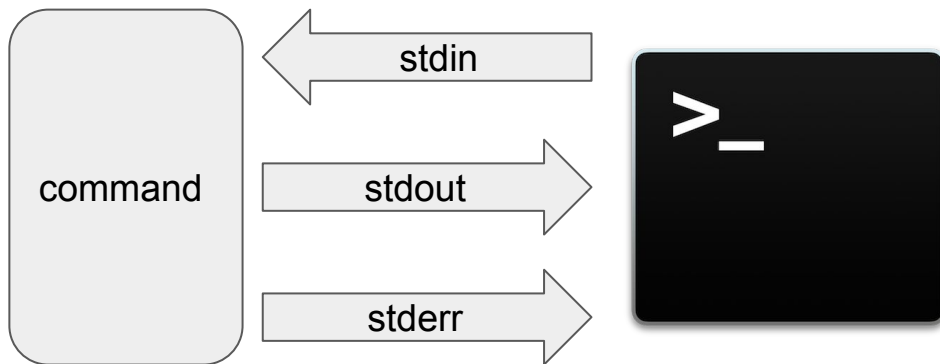
Verranno visualizzati **sul terminale** i nomi dei file presenti nella directory, tra cui 10 file in formato FASTA contenenti sequenze di trascritti riconducibili a geni coinvolti nel cancro al seno.



Flusso di dati nella shell: stdin, stdout, stderr

stdin, stdout, and stderr are three data streams created when you launch a Linux command

Error messages from the command are sent through the stderr (standard error) stream



Formato di file FASTA

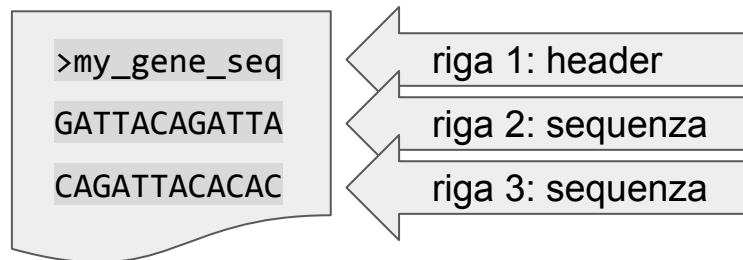
Il formato FASTA deriva dal programma di allineamento di sequenze di DNA e proteine “FASTA” (FAST-All). Esistono anche FASTP (proteins) e FASTN (nucleotides).

FASTA is a text-based format for representing either nucleotide sequences or amino acid (protein) sequences, in which nucleotides or amino acids are represented using single-letter codes

Un file in formato FASTA contiene una riga di intestazione che inizia con il carattere > seguito dal nome della sequenza (e.g. il nome del gene), le righe sottostanti l'intestazione sono la sequenza.

Solitamente hanno nomi del file con estensione .fasta o .fa

Un esempio minimale di una sequenza in formato FASTA:



Multi-FASTA file

Un file contenente molteplici sequenze in formato FASTA è detto multi-FASTA.

my_multifasta.fa

```
>gene1_seq  
GATTACAGATTA  
CAGATTACACAC
```

```
>gene2_seq  
CAGATTACA
```

```
>gene3_seq  
NACAGATTACA
```

Ogni sequenza FASTA ha la propria intestazione, quindi sequenze di entità (geni, trascritti, proteine, ...) diverse saranno riconoscibili.

Le sequenze genomiche sono disponibili nei database in file FASTA separati, uno per cromosoma, sia in un unico file in cui le sequenze dei cromosomi sono state concatenate.

Vedi <http://www.ensembl.org/info/data/ftp/index.html>

Visualizzare sul terminale (grandi) file di testo

Visualizzare il contenuto di uno dei file con il comando “less”:

```
$ man less
```

“Less does not have to read the entire input file before starting, so with large input files it starts up faster than text editors”

```
$ less seq01.fasta
```

Usare le frecce verso l’alto e verso il basso per scorrere il contenuto del file.

Per uscire dalla visualizzazione e tornare al prompt dei comandi premere il tasto **q** (“quit”).

```
$ less -S seq01.fasta
```

-S or --chop-long-lines
Causes lines longer than the screen width to be chopped (truncated) rather than wrapped.

Creare un multi-FASTA: concatenare diversi file

Useremo il comando **cat** (concatenate) per concatenare due file FASTA (file di testo).

```
$ man cat
```

“cat [OPTION]... [FILE]...

cat - concatenate files and print on the standard output”

```
$ cat seq01.fasta seq02.fasta
```

Creare un multi-FASTA: concatenare diversi file

Useremo il comando **cat** (concatenate) per concatenare due file FASTA (file di testo).



Not me(ow)!

```
$ man cat
```

“cat [OPTION]... [FILE]...

cat - concatenate files and print on the standard output”

```
$ cat seq01.fasta seq02.fasta
```

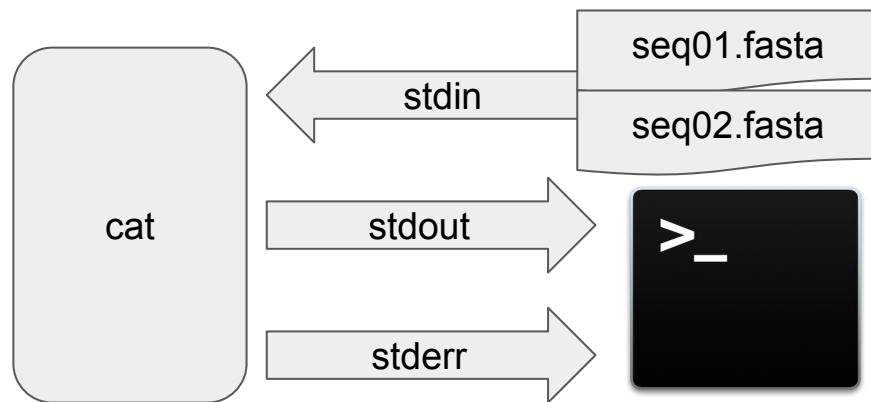

Creare un multi-FASTA: concatenare diversi file

“cat [OPTION]... [FILE]...”

cat - concatenate files and print on the standard output”

```
$ cat seq01.fasta seq02.fasta
```

Vedremo le due sequenze stampate nel terminale poichè lo stdout di cat è indirizzato al terminale



Creare un file multi-FASTA: l'operatore >

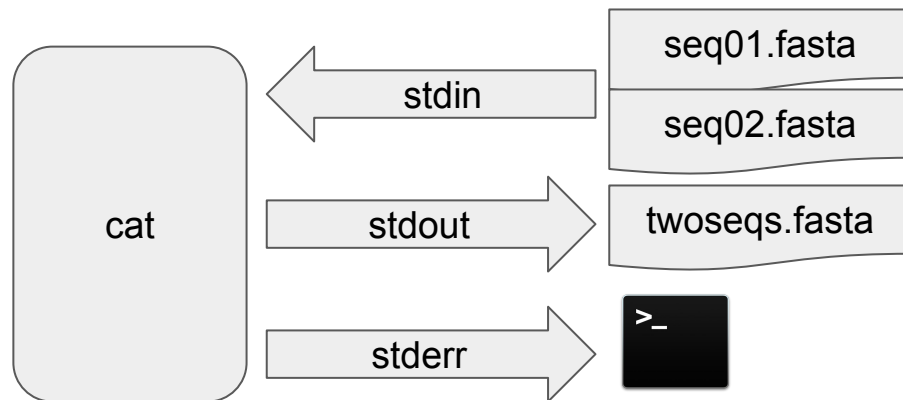
Per scrivere un nuovo file ci serviremo dell'operatore di redirectione dello standard output ">"

```
$ cat seq01.fasta seq02.fasta > twoseqs.fasta
```

">" crea un nuovo file o sovrascrive un file esistente!

N.B.: non confondere l'operatore di redirectione con il carattere indicatore di intestazione FASTA !

Il carattere ">" è il medesimo, ma i contesti sono diversi.



Creare un file multi-FASTA: concatenare file multipli

Creare un unico file FASTA a partire da singole sequenze, un file per sequenza: concateniamo tutte le sequenze FASTA in un unico file

```
$ cat seq01.fasta seq02.fasta [...] seq10.fasta > tenseqs.fasta
```

Una scorciatoia sfruttando le caratteristiche della shell:

usiamo un carattere “wild-card”, cioè un carattere speciale che sta ad indicare qualsiasi carattere (anche nessuno), ***** (l’asterisco)

```
$ cat seq*.fasta > all.fasta
```

seq*.fasta viene “espanso” ai nomi dei file che iniziano con “seq”, contengono 0 o più caratteri e poi finiscono con “.fasta”

(nelle esercitazioni successive capiremo meglio questo comportamento, quando vedremo le “espressioni regolari”)

Il mio primo multi-FASTA

Verifichiamo che il file appena creato contenga le sequenze dei vari file

```
$ less all.fasta
```

Abbiamo creato un file multi-FASTA !



Appendere file

E se ci fossimo dimenticati di aggiungere un file alla nostra catena?

L'operatore ">>" serve ad appendere l'output del comando a sinistra alla fine del file a destra

```
$ cat coli_genes.fasta >> all.fasta
```

Verifichiamo l'output

```
$ less all.fasta
```

Premendo "G" maiuscolo (Shift+g) less ci porta alla fine del file.

Per tornare all'inizio "g" (minuscolo)

Ottenere l'inizio e la fine di un file

Per visualizzare le prime N righe di un file

```
$head all.fasta
```

Si può scegliere quante righe

```
$head -n 20 all.fasta
```

 (

```
$head -20 all.fasta
```

)

Per visualizzare le ultime N righe di un file

```
$tail all.fasta
```

anche per tail si possono scegliere quante righe

(contando a partire dal fondo)

```
$tail -n 22 all.fasta
```



Ottenere statistiche di un file di testo

Contare il numero totale di righe presenti nel file all.fasta:

utilizzare il comando wc

```
$ wc all.fasta
```

```
516 536 30936 all.fasta
```

```
$ man wc
```

*“wc - print **newline**, **word**, and **byte** counts for each file”*

Alcune opzioni interessanti:

-l (lines)

```
$ wc -l all.fasta
```

```
516 all.fasta
```

-w (words)

```
$ wc -w all.fasta
```

```
536 all.fasta
```

-c (bytes)

```
$ wc -c all.fasta
```

```
30936 all.fasta
```

-m (characters)

```
$ wc -m all.fasta
```

... è simile al numero di bytes?

Ottenere statistiche di un file di testo

Contare il numero totale di righe presenti nel file all.fasta:
utilizzare il comando wc

```
$ wc all.fasta
```

```
516 536 30936 all.fasta
```

```
$ man wc
```

*“wc - print **newline**, **word**, and **byte** counts for each file”*



Kitty!
“wc” stands for
“word count”!

Alcune opzioni interessanti:

-l (lines)

```
$ wc -l all.fasta
```

```
516 all.fasta
```

-w (words)

```
$ wc -w all.fasta
```

```
536 all.fasta
```

-c (bytes)

```
$ wc -c all.fasta
```

```
30936 all.fasta
```

-m (characters)

```
$ wc -m all.fasta
```

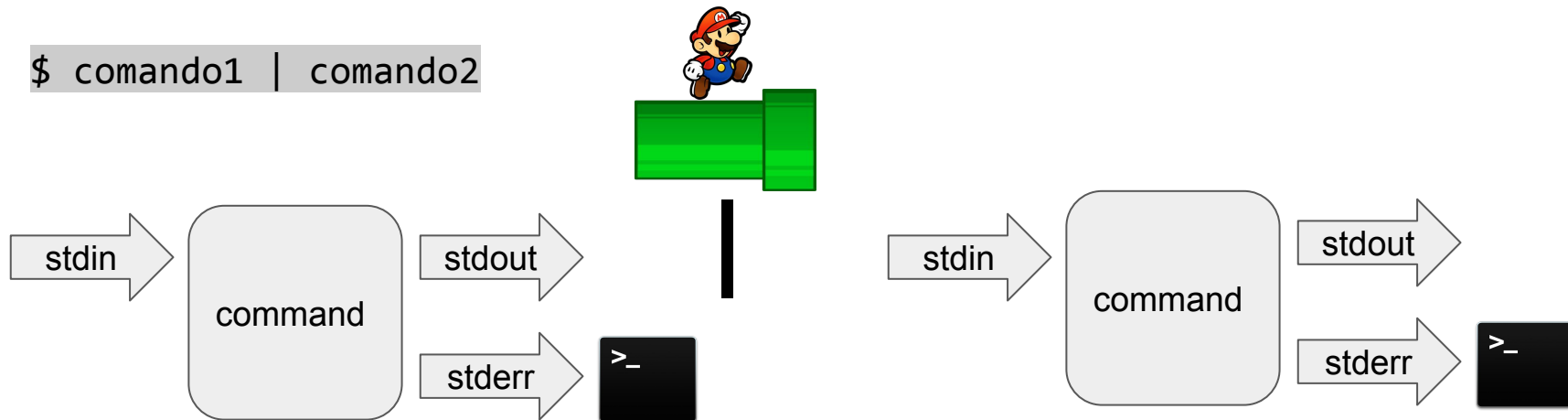
... è simile al numero di bytes?

L'operatore "pipe"

Con la shell Linux è possibile passare l'output di un comando direttamente ad un altro comando senza dover scrivere un file intermedio.

L'operatore | (pipe o barra verticale) collega l'output del comando a sinistra con l'input del comando alla sua destra:

```
$ comando1 | comando2
```



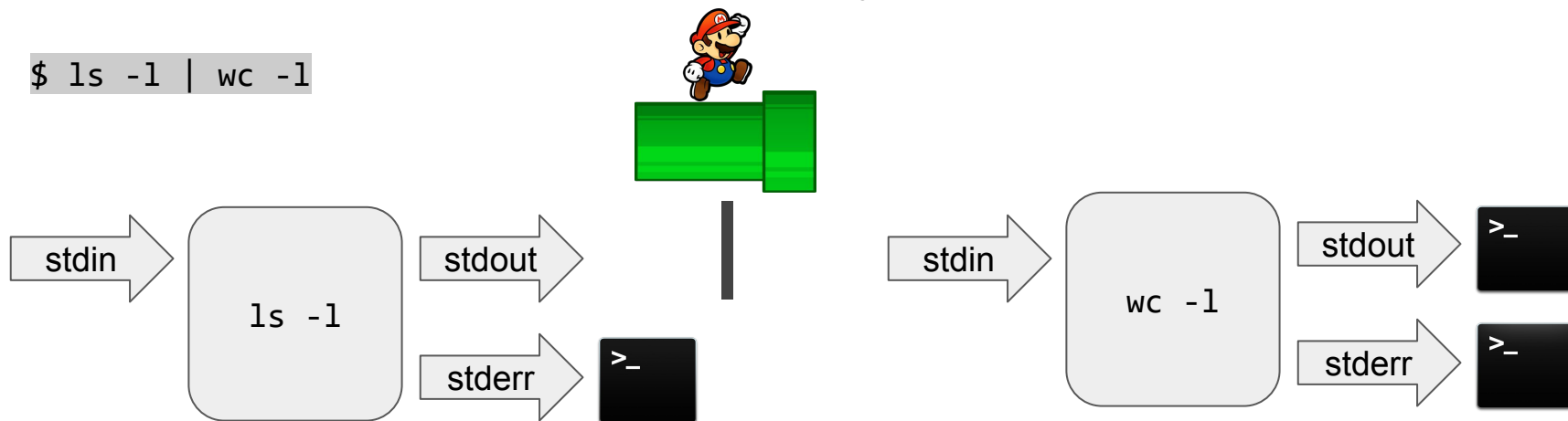
L'operatore “pipe”

Ad esempio, un altro modo per contare il numero di righe nel file all.fasta è:

```
$ cat all.fasta | wc -l
```

Oppure, contare quanti file ci sono nella directory

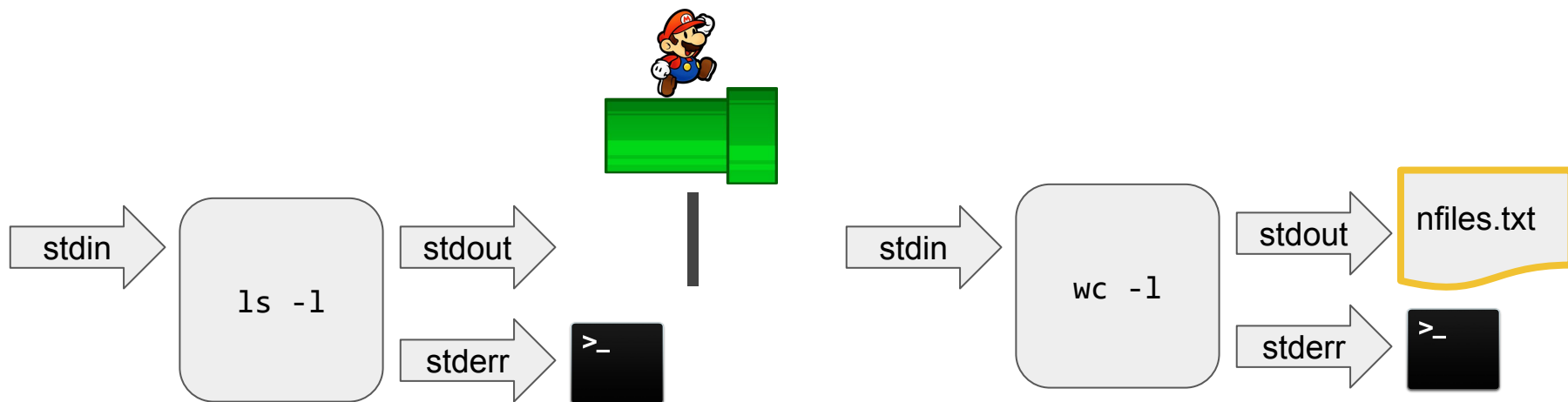
```
$ ls -l | wc -l
```



Una semplice pipeline di comandi

Contare quanti file ci sono nella directory e salvare il risultato in un file

```
$ ls -l | wc -l > nfiles.txt
```



Virtual Machines

The image depicts a perspective view of a digital corridor. The walls, floor, and ceiling are composed of a dense, green, pixelated pattern of binary code (0s and 1s). Several rectangular frames, glowing with a bright green light, are positioned along the walls and floor, receding into the distance. The overall effect is a sense of depth and immersion in a virtual, data-driven environment.

<https://vlab.vdi.ict.unipd.it>