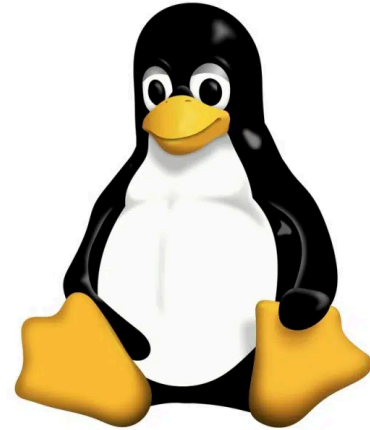


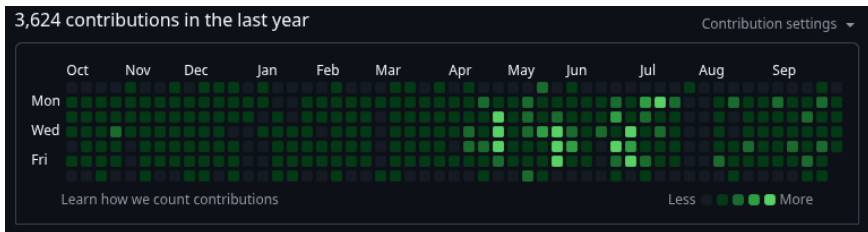
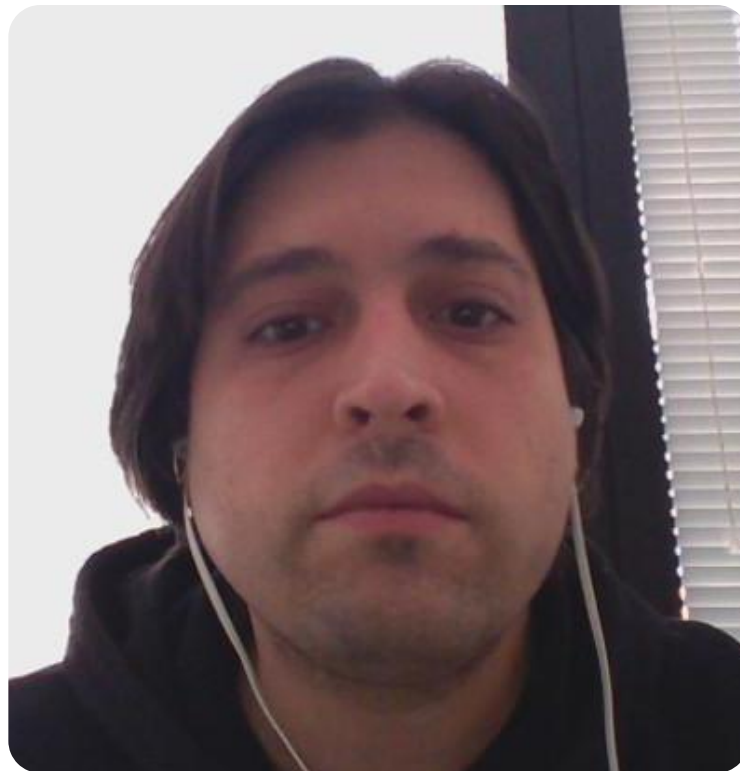
# Unix Philosophy

Linux day 2025



# Who am I?

- Ingegnere e architetto software  
([platformatic.dev](https://platformatic.dev))
- Imprenditore ([remotamente.it](https://remotamente.it))
- Consulente freelance
- Maker
- Linux user dagli anni '90



# Un po' di (mia) storia

- Commodore 64 (1985)
- Amiga 500 (1988)
- i486 / Pentium (1991/1998)
- Circa 1996: linux
- 1999: abbandono definitivamente Windows per RH Linux
- Currently: POP!\_OS / Debian



# Storia di Linux

- 25/8/1991: Linus Torvalds annuncia che sta lavorando a un sistema operativo libero per i386.
- 1992: X Window System su linux
- 13/3/1994: Kernel 1.0
- Il Kernel oggi

*NOTA:* Quello che chiamiamo "linux" è in realtà il kernel linux + una collezione di software libero chiamato GNU. Spesso si indica come GNU/Linux



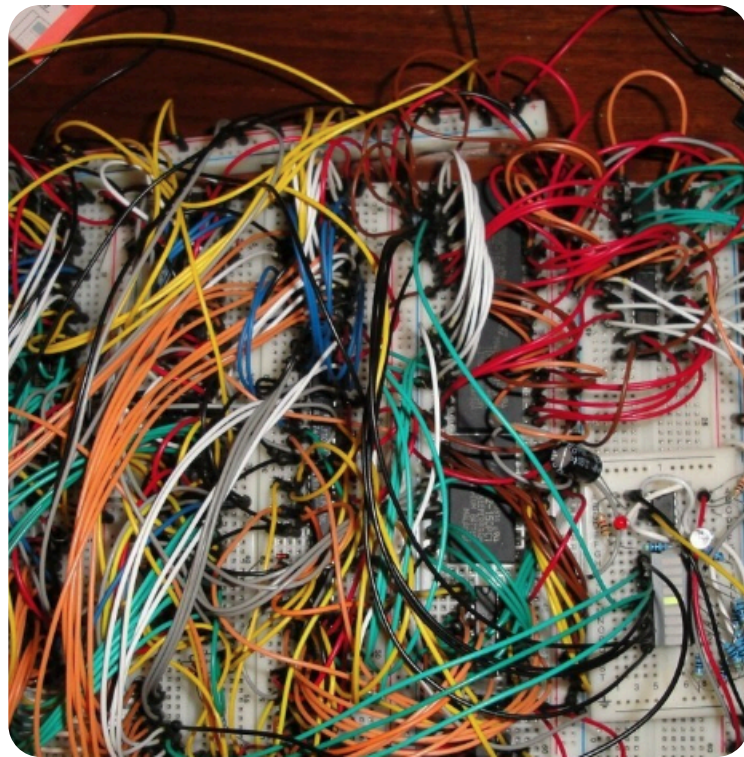


"Unix is simple. It just takes a genius to understand its simplicity."

Dennis Ritchie

# Complessità

- La complessità è un grosso problema nel mondo del software.
- Software "complesso" è difficile da capire e da mantenere.
- La filosofia di Unix enfatizza la costruzione di codice semplice, compatto, chiaro, modulare e estensibile che può essere facilmente mantenuto da sviluppatori diversi rispetto ai creatori originali
- In generale, si favorisce la composability rispetto a design monolitici



# KISS Principle

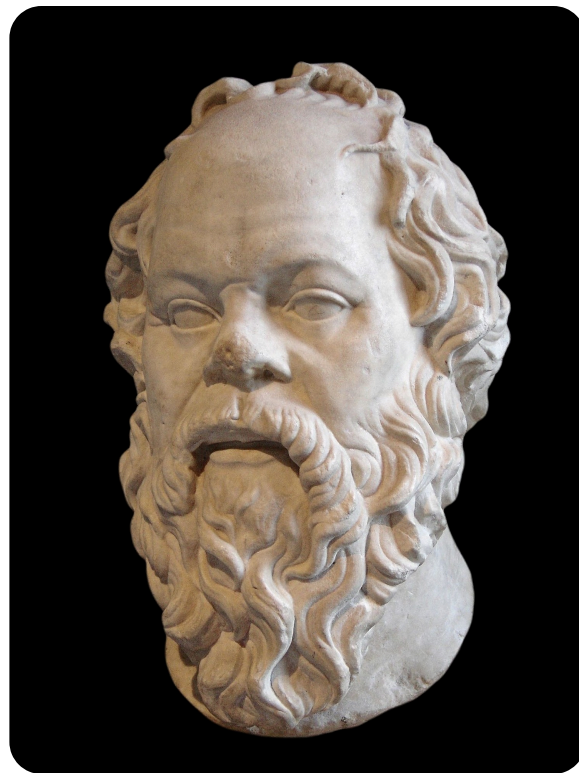
Keep it simple, stupid!



# Unix Philosophy - Doug McIlroy (1978)

Doug McIlroy nel Bell System Technical Journal (1978):

- Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new "features".
- Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.
- Design and build software, even operating systems, to be tried early, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.
- Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.



# Unix Philosophy - Peter H. Salus (1994)

Peter H. Salus in "A Quarter-Century of Unix" (1994) riassume:

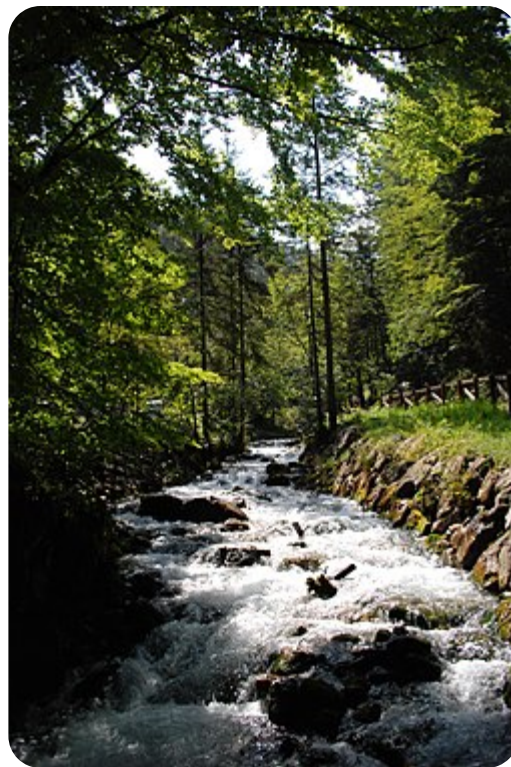
- Write programs that do one thing and do it well
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

Questa è la versione più concisa e citata.



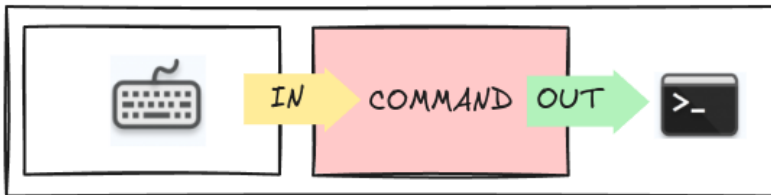
# Esempio: Filtri e streams

- Un filtro software è un programma che processa uno *stream* e produce un altro *stream*.
- I filtri possono essere usati individualmente, ma sono spesso combinati in "pipelines"



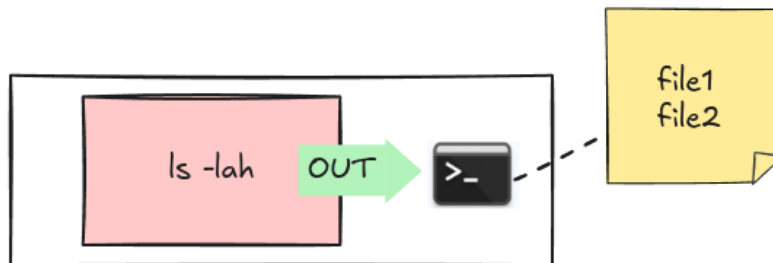
# Filtri Unix

- Ogni processo lanciato su (emulazione di) terminale può:
  - Ricevere input da ``stdin`` (standard input)
  - Scrivere output su ``stdout`` / ``stderr`` (standard output/error)
- Questo è il motivo per cui quando lanciate un programma da terminale, vedete il suo output sullo schermo, il terminale prende stdout e lo stampa.
- Funziona perché ogni processo ha una "file descriptor table" dove 0, 1 e 2 sono mappati su stdin, stdout, stderr, rispettivamente.



# Comandi senza input

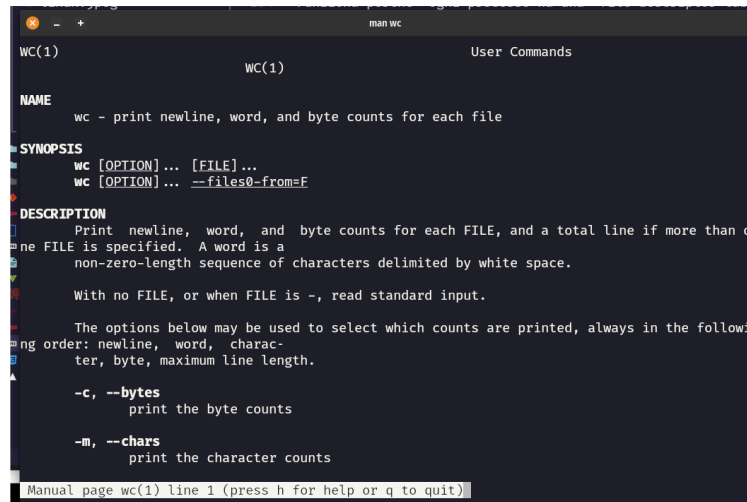
- Alcuni comandi agiscono "senza input" ma scrivono comunque su ``stdout`` eg: ``ls -lah``



# "Do one thing and do it well"

Alcuni dei filtri più usati:

- wc
- grep
- sort
- head / tail
- cat
- shuf
- rev
- more
- tee



```
man wc
WC(1)                                WC(1)                                User Commands

NAME
wc - print newline, word, and byte counts for each file

SYNOPSIS
wc [OPTION]... [FILE]...
wc [OPTION]... --files0-from=F

DESCRIPTION
Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified. A word is a non-zero-length sequence of characters delimited by white space.

With no FILE, or when FILE is -, read standard input.

The options below may be used to select which counts are printed, always in the following order: newline, word, character, byte, maximum line length.

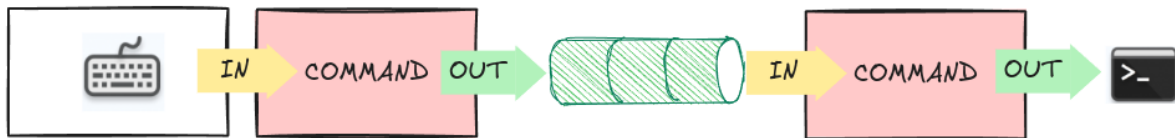
-c, --bytes
    print the byte counts

-m, --chars
    print the character counts

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

# "Work together and handle streams"

- Le pipe permettono ai dati di un processo di essere passati ad un altro tramite un flusso unidirezionale.
- I filtri unix possono anche essere concatenati con le "pipe" (|): lo ``stdout`` di un processo diventa l'input (``stdin``) di quello successivo.

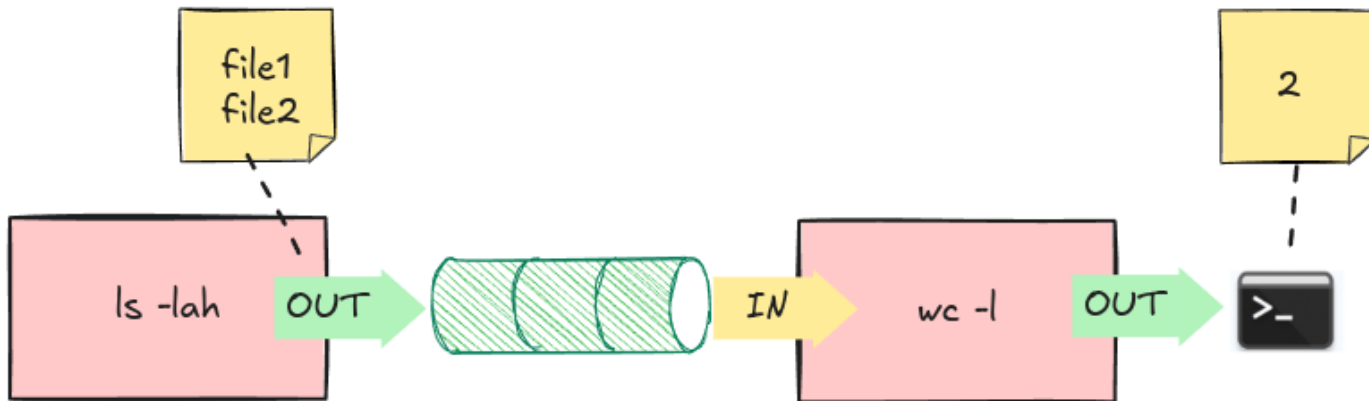


In ``bash`` :

```
comando_1 | comando_2 | comando_3 | .... | comando_N
```

# Quanti file ho nel folder?

```
ls | wc -l
```

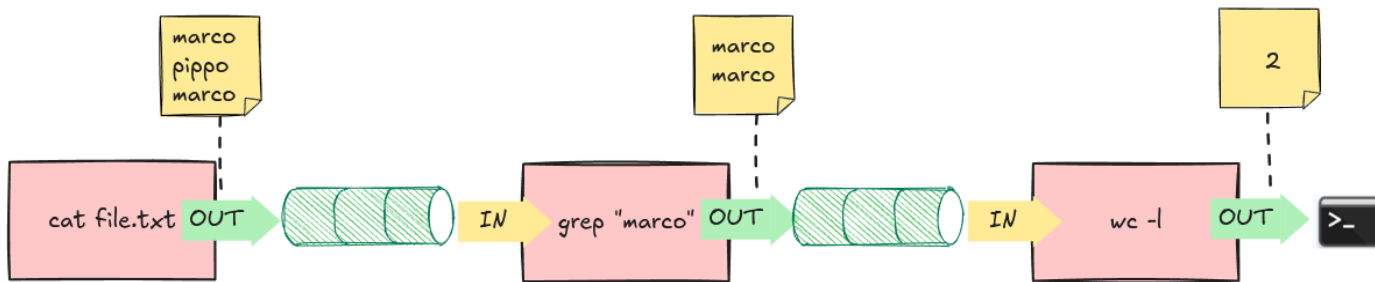




# Quante volte un file contiene una parola?

- ``cat`` riproduce il file su ``stdout``
- ``grep`` filtra le righe che contengono il parametro
- ``wc -l`` le conta

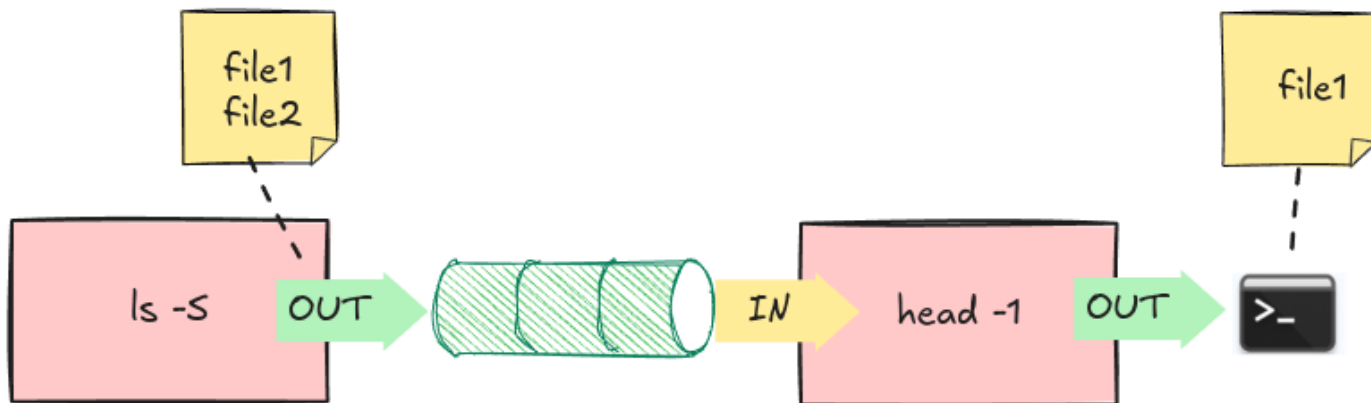
```
cat file.txt | grep "marco" | wc -l
```



# Troviamo il file più grande in un folder

- ``ls -S`` : ordina per dimensione, il più grande per primo, un file per riga (se c'è un |)
- ``head -1`` : prende il primo elemento

```
ls -S | head -1
```



# Dado-Mucca a sei facce

- ``seq`` stampa una sequenza di numeri
- ``shuf`` li mischia
- ``head -n 1`` prende l'inizio di ``stdout`` (la prima riga)
- ``cowsay`` è la mucca
- ``lolcat`` colora

```
seq 1 6 | shuf | head -n 1 | cowsay | lolcat
```

...provare anche con ``figlet``



# Mucca - versione script bash

Possiamo comporre tutto in uno script ``bash``, in modo tale da poterlo lanciare direttamente

Non dimenticate di impostare i diritti di esecuzione ( ``chmod`` )

```
#!/bin/bash  
seq 1 6 | shuf | head -n 1 | cowsay
```

...o usando ``$1`` per passare un parametro

```
#!/bin/bash  
seq 1 $1 | shuf | head -n 1 | cowsay
```

# Aggiungiamo la voce!

POSSIAMO USARE ESPEAK:

```
espeak -v it test
```

...CHE OVVIAMENTE È UN FILTRO

```
echo "marco" | espeak -v it
```

DADO PARLANTE:

```
seq 1 6 | shuf | head -n 1 | espeak -v it
```

# Stream from youtube

Linus torvalds introduces Linux 1.0 (1994): <https://www.youtube.com/watch?v=qaDpjIFpbfo>

- ``yt-dlp -o -`` : scrive su stdout
- ``mplayer -`` : legge da stdin

```
yt-dlp -q -o - 'https://www.youtube.com/watch?v=qaDpjIFpbfo' | mplayer -
```

Versione script:

```
#!/bin/bash  
yt-dlp $1 -q -o - | mplayer -
```

# Altri esempi interessanti

TOP 10 COMANDI PIÙ USATI DALLA HISTORY

```
history | awk '{print $2}' | sort | uniq -c | sort -rn | head -10
```

- ``awk '{print $2}'`` : estrae il secondo campo (il comando)
- ``uniq -c`` : conta le occorrenze
- ``sort -rn`` : ordina numericamente in modo inverso

# Password casuale

```
tr -dc A-Za-z0-9 < /dev/urandom | head -c 16 | cowsay
```

- `/dev/urandom` : genera dati casuali
- `tr -dc A-Za-z0-9` : filtra solo lettere e numeri
- `head -c 16` : prende i primi 16 caratteri



# Previsioni meteo nel terminale

```
curl wttr.in/Bologna | lolcat
```

- ``curl wttr.in/Bologna`` : scarica le previsioni meteo per Bologna
- ``lolcat`` : colora l'output

# ASCII art

```
figlet "Linux Day" | lolcat
```

o combinato con ``cowsay`` :

```
figlet "Linux Day" | cowsay -n | lolcat
```

- ``figlet`` : converte testo in ASCII art
- ``cowsay -n`` : la mucca dice il testo (senza bubble)

# Grazie!

## CONTATTI

- LinkedIn
- Twitter
- GitHub
- Email

## RIFERIMENTI

- Queste slides
- The Unix Philosophy
- The GNU Manifesto
- sli.dev