

Il terminale

Storia di come il terminale moderno è divenuto a essere, dalla telescrivente allo schermo, dal TTY all'emulatore di terminale

Chi sono

$ whoami
marco_rondelli

Capitoli

1. Metodi di comunicazione
    1. Cosa è un automa
    2. Cosa è un input
    3. Cosa è un output
    4. La telescrivente
2. Il terminale
    1. Il terminale del passato
    2. Il terminale del presente
    3. L’emulatore di terminale
3. Come accedere al terminale
4. Gestire terminali simultaneamente
5. FAQ

1. Metodi di comunicazione

« Ai miei tempi era tutto più semplice… »

Ne siamo sicuri?
Scopriamolo assieme!

$ timg zinturre.jpg



$ timg *.gif



$ timg backintime.jpg



Marca: IBM
Modello: 1401
Bits: 6 variabile (8)
Endianness: Big-endian
Frequenza: 87kHz

Peso (CPU): ~900kg
Peso (lettore): ~610kg
Peso (stampante): ~1450kg
Peso (nastri): ~450kg

$ date “+%Y-%m-%d”

1960-01-01

$ timg ibm1401.jpg




WE CAN NOW RUN OUR COMPILED CODE
JUST PRESS START...

1.1 Cosa è un automa

```
 _____________________________________
< Un automa è una scatola misteriosa che >
<                    mangia qualcosa per                    >
<                    produrre qualcosa              >
 -------------------------------------
                                                \  ^__^
                                                 \ (oo)\_______
                                                   (__)\       )\/\
                                                       ||----w |
                                                       ||     ||
```

Una definizione formale di automa esiste da oltre 60 anni. Usiamo quella per capire cosa è un automa, come parlarci e come ottenere informazioni da esso.

NOTA: capire cosa è un automa ci permette di comprendere il perché certi meccanismi delle macchine esistono e come funzionano.

_____

< Un automa è rappresentato da una quintupla $M = \langle \Sigma, \Gamma, Q, \delta, \lambda \rangle$, dove: >
< Σ è un insieme di simboli chiamato alfabeto input, >
< Γ è un insieme di simboli chiamato alfabeto output, >
< Q è un insieme di stati (finito), >
< δ è una funzione di transizione $\delta: Q \times \Sigma \rightarrow Q$, >
< λ è una funzione di transizione $\lambda: Q \times \Sigma \rightarrow \Gamma$ >

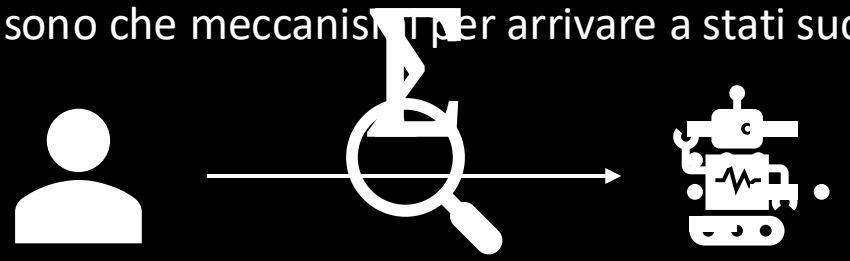
Q: Come parliamo a un computer?
A: Con l'alfabeto input.

Q: Come ci risponde il computer?
A: Con l'alfabeto output.

Q: In che stato è il computer?
A: In uno degli stati dell'insieme.

```
\  ^__^
 \ (oo)\_______
   (__)\       )\/\
      ||----w |
      ||     ||
```

Le funzioni di transizione altro non sono che meccanismi per arrivare a stati successivi, dati stati intermedi e output.



Persona Automa

1.2 Cosa è un input

Un *alfabeto finito* di *simboli* che l'automa è capace di interpretare.

0123456789ABCDEFGHIJKLMNOPQRSTWXYZ

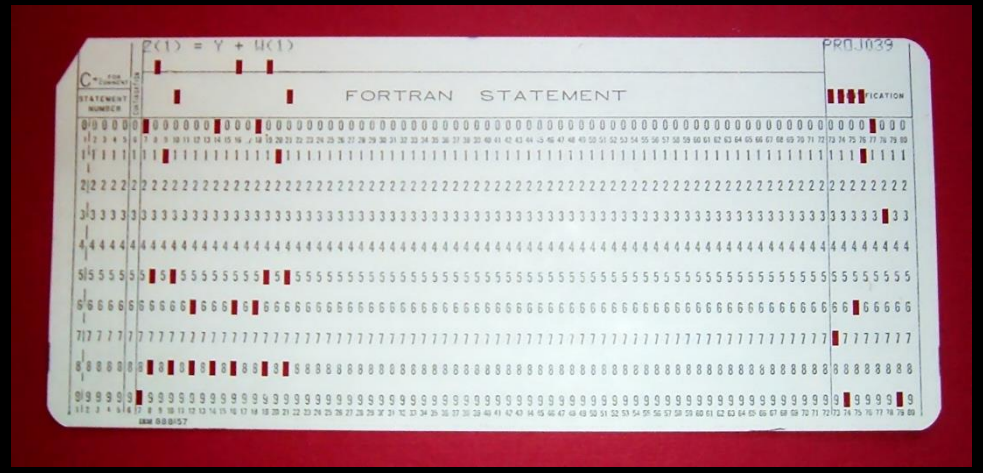
Tramite *circuiti logici* interni, la macchina interpreta i simboli in ingresso, verificando che questi siano simboli validi, davvero esistenti nell'alfabeto.

Negli anni 60', esistono due tipi principali di input.

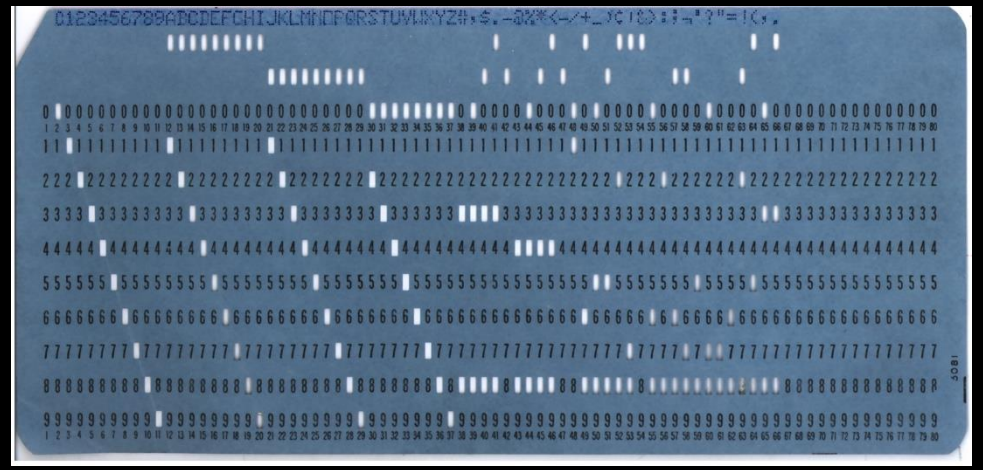
- Scheda perforata
- Nastro perforato

La macchina responsabile della creazione di queste schede e nastri si chiama perforatrice ed è operata manualmente.

$ timg fortran00.jpg



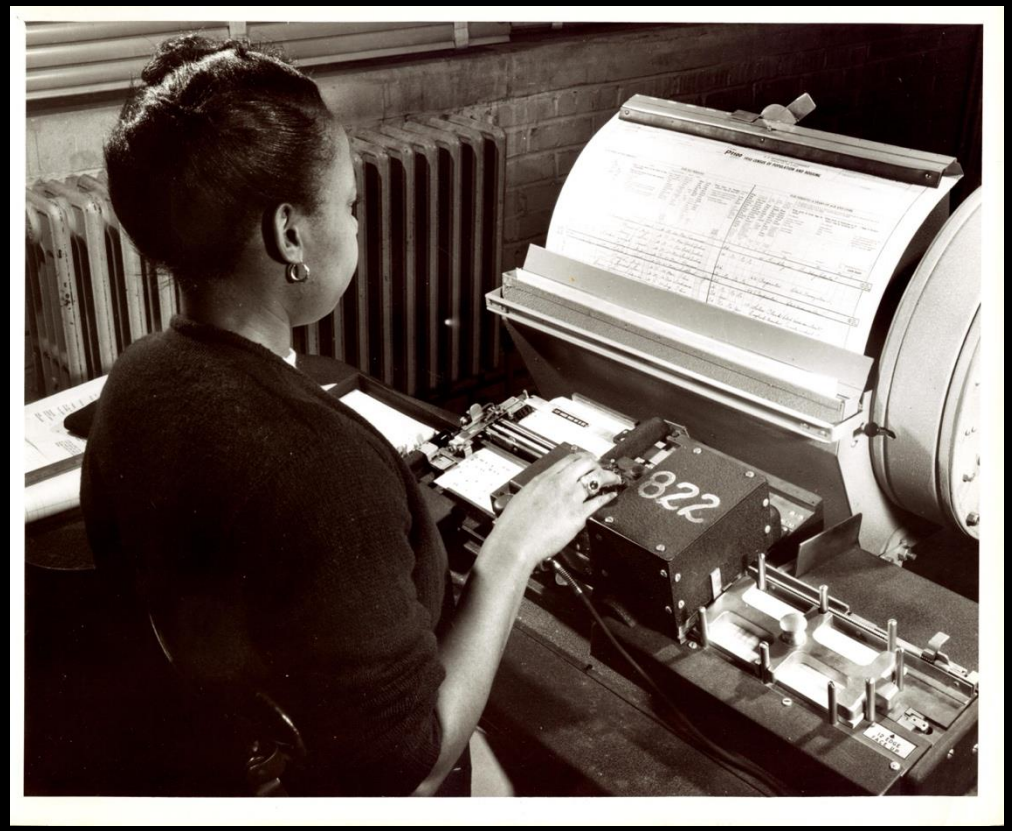
$ timg alphabet.jpg



$ timg card_puncher00.jpg



$ timg card_puncher01.jpg



1.3 Cosa è un output

L’automa interpreta l’alfabeto e dopo avere eseguito delle operazioni interne, produce un output.

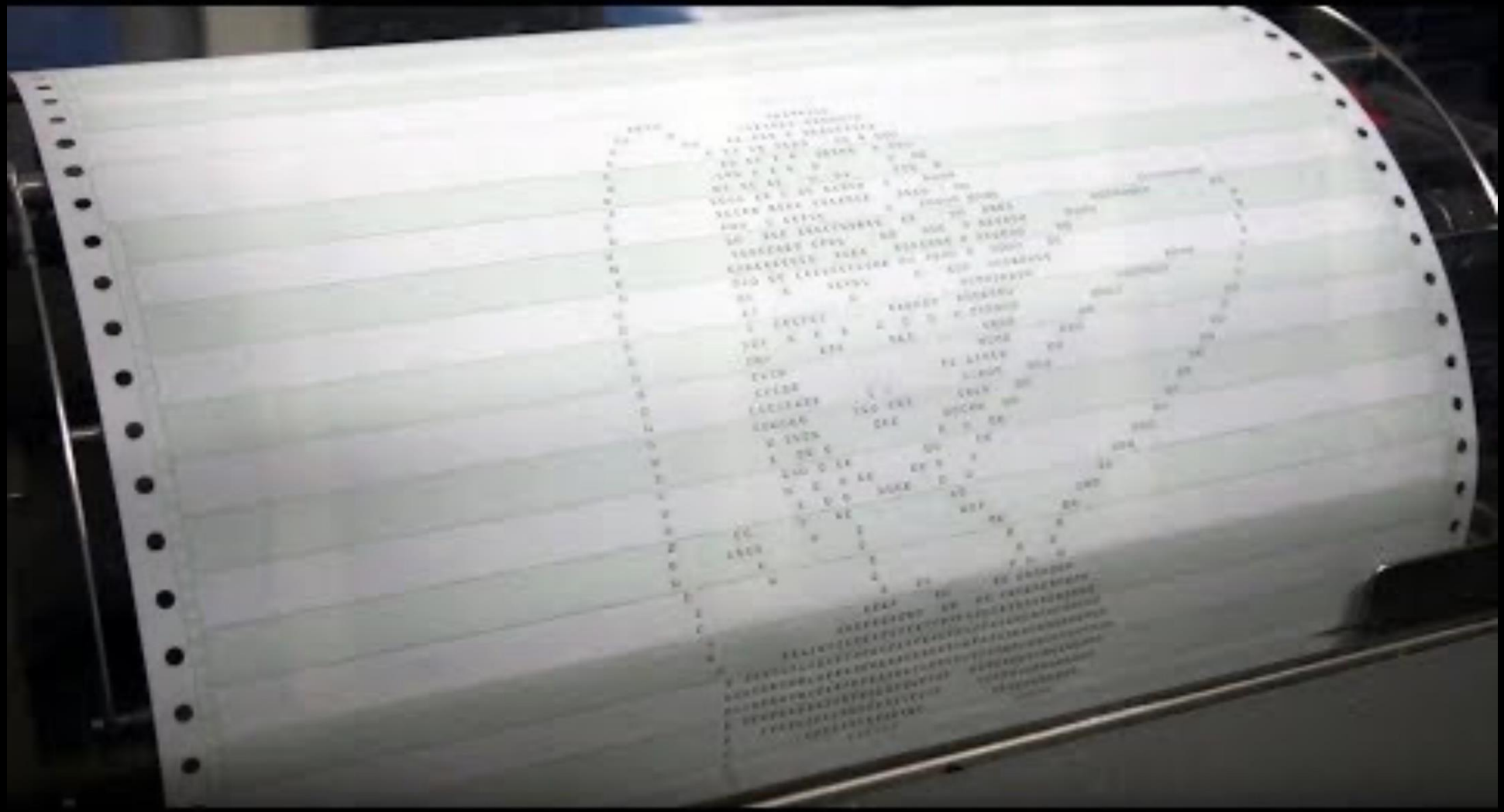
L’output può essere segnalato tramite suoni, luci, o comunque segnali elettromagnetici in principio.

L’output può essere interpretato nello stesso modo in cui è stato interpretato l’input, usando lo stesso alfabeto di simboli.

La macchina responsabile di segnalare l’output all’esterno è chiamata stampante.

$ timg ibm_line_printer.jpg





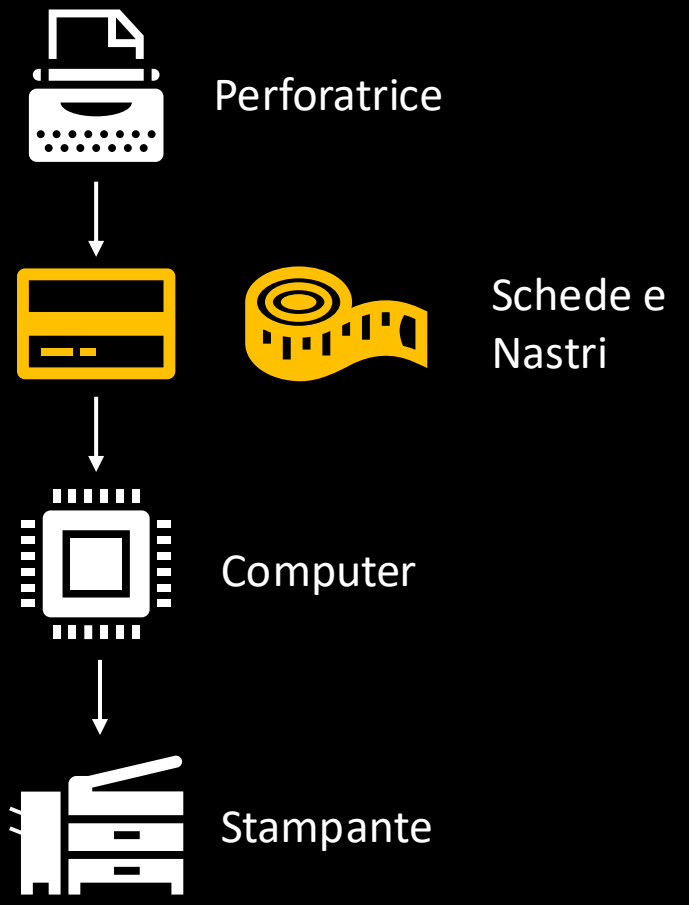
1.4 La telescrivente

I metodi di input e output che abbiamo visto sono stati lo standard di comunicazione con le macchine per molti anni.

Similmente, i nastri magnetici usano il concetto di scrittura su nastro, fornendo un'alternativa ad alta densità, ma comunque incredibilmente lenta.

L'uso della carta perforata non era una complicazione, bensì una soluzione alle problematiche del tempo.

- Limiti nella velocità di input.
- Limiti nella comunicazione.
- Limiti nella memoria.
- Limiti nella conservazione.

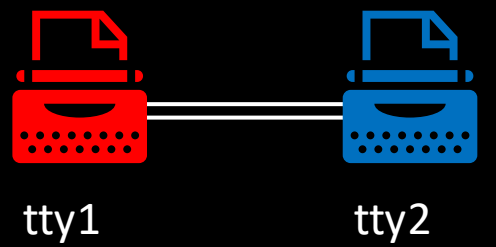


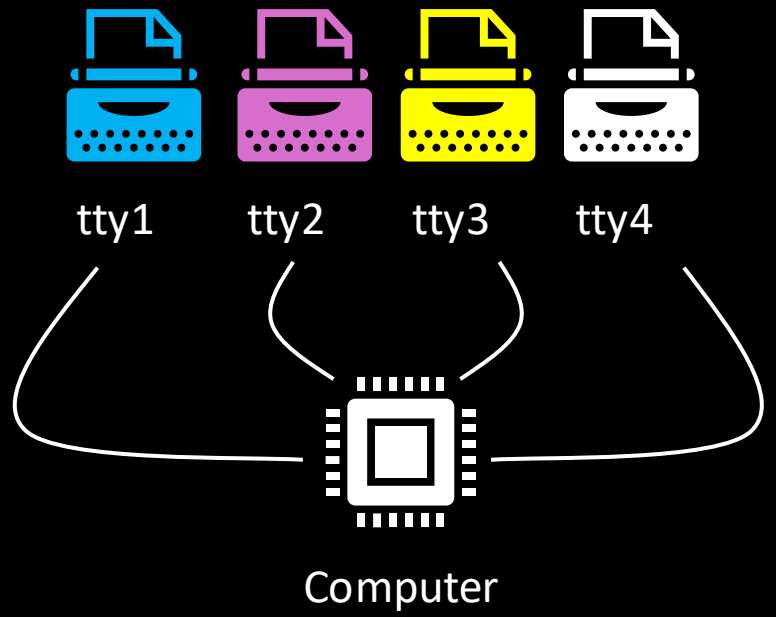
Le telescriventi (TTYs) esistevano già da decenni, ideate come mezzi di comunicazione a distanza, facenti uso del *telegrafo* e *linee telefoniche* come mezzo di trasporto.

La diminuzione dei costi di produzione degli schermi a tubo catodico (CRT), ma soprattutto, l'avvento dei sistemi di time-sharing, furono gli eventi che permisero ai videoterminali di prendere il sopravvento.

Tramite la condivisione del tempo al calcolatore, una moltitudine di persone ha accesso simultaneo. Il calcolatore diventa responsabile della suddivisione automatica di risorse.

I vantaggi superano i costi.

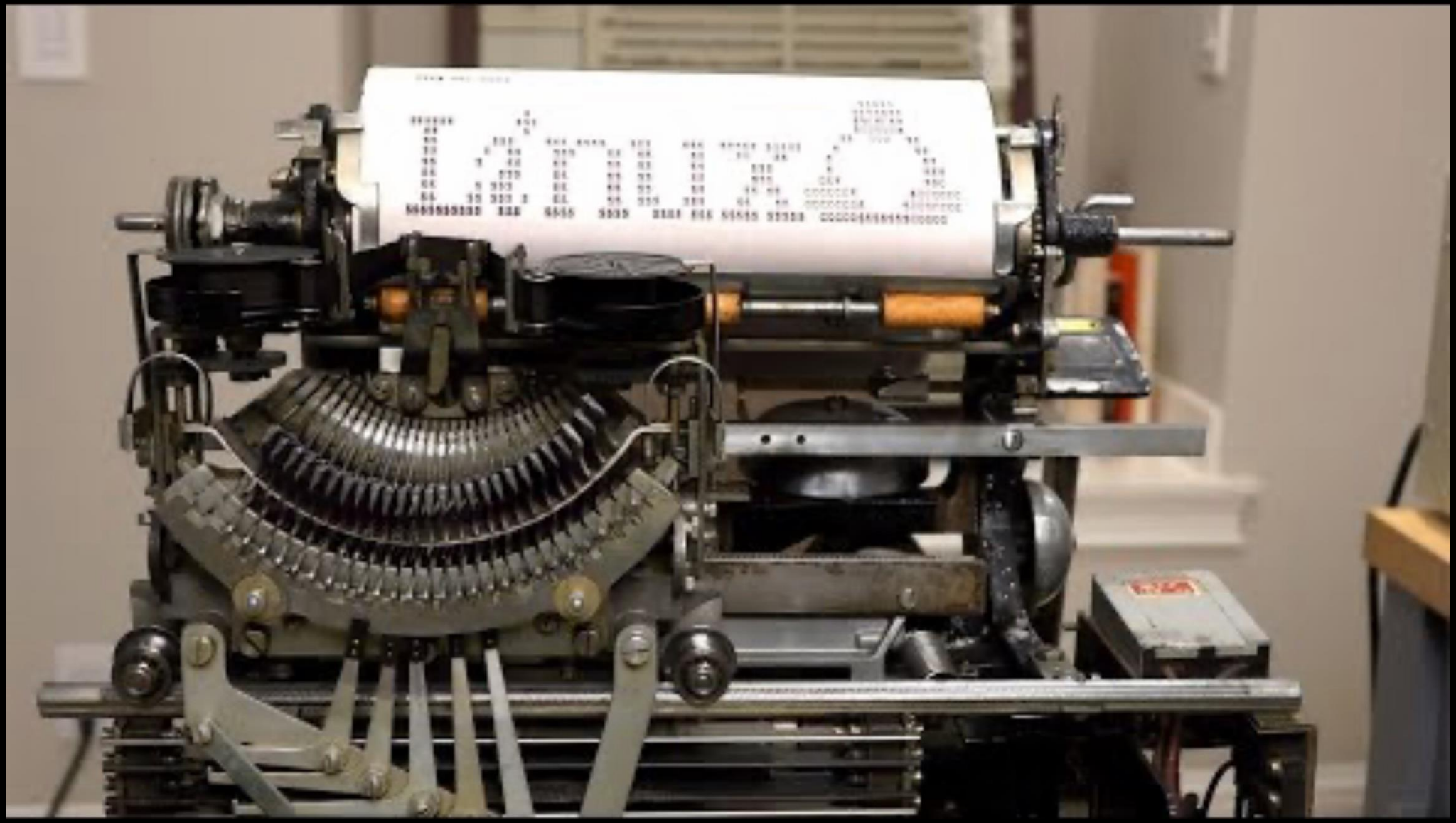


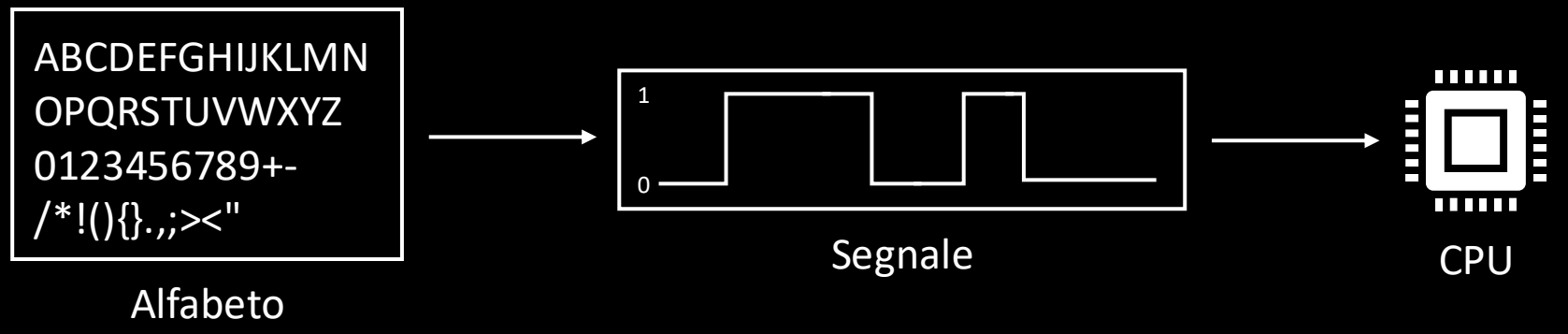


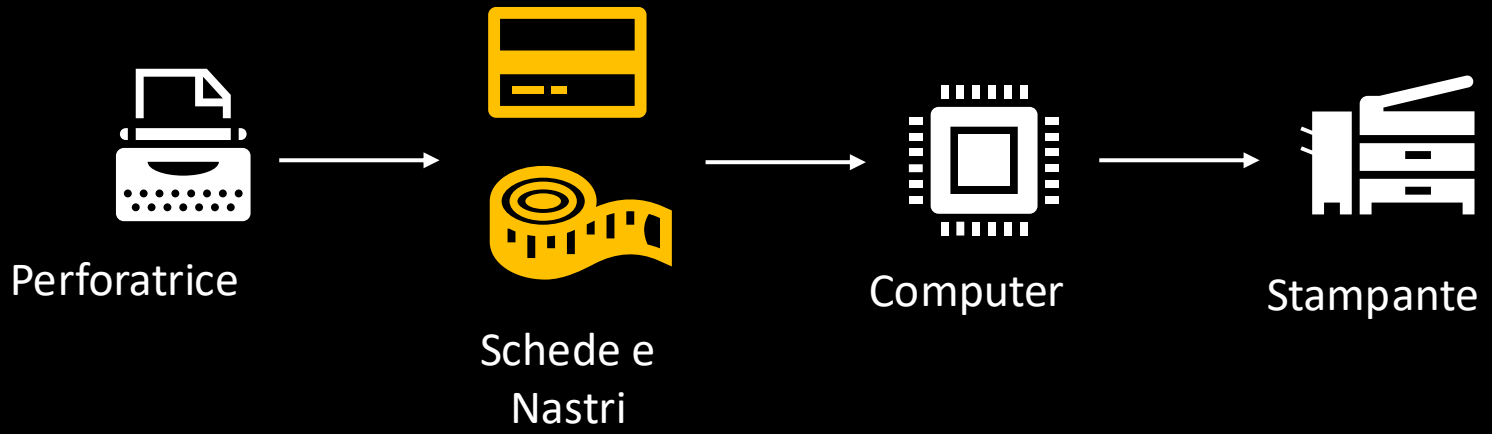
$ timg tty33.jpg

$ timg tty13.jpg






ABCDEFGHIJKLMN
OPQRSTUVWXYZ
0123456789+-
/*!(){}.,;><"
Alfabeto
1
0
Segnale
CPU


Perforatrice
Schede e
Nastri
Computer
Stampante

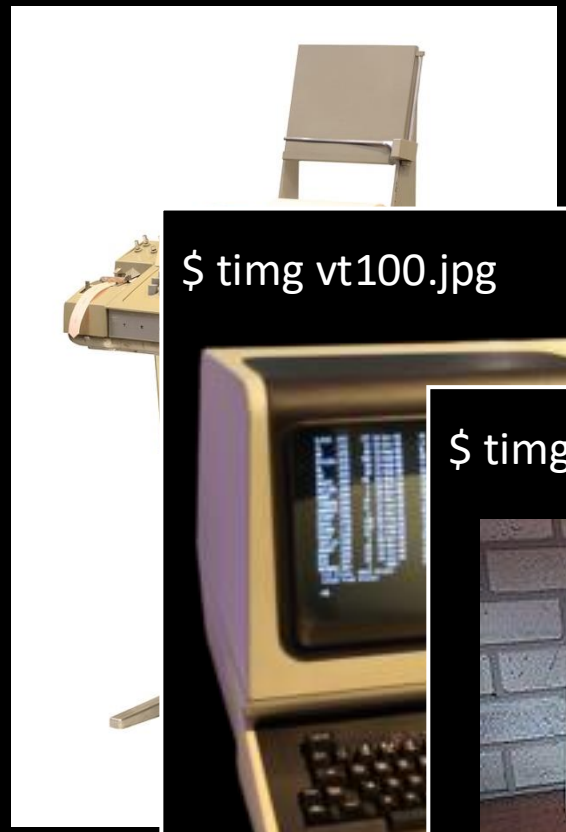

TTYs
Computer

2. Il terminale

$ timg vt100.jpg

$ timg terminal.jpg

$ timg tty_mod_33.jpg

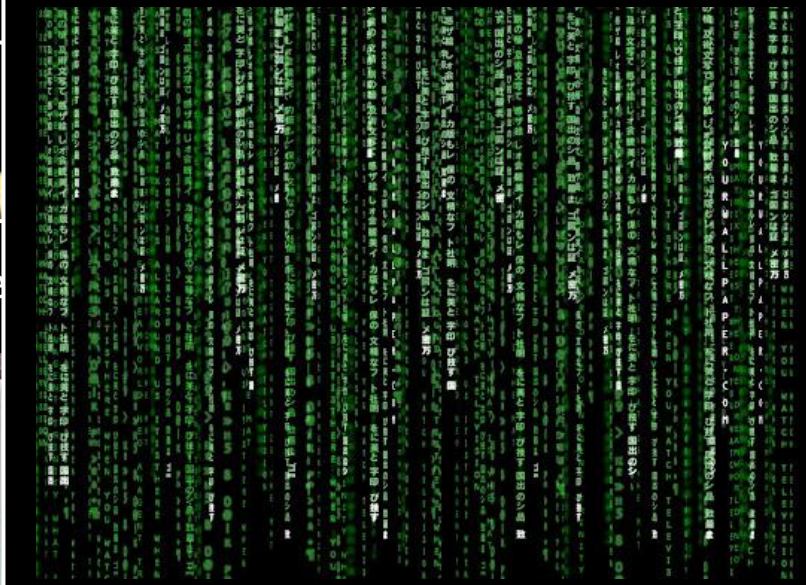


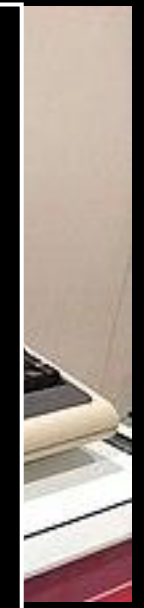
$ timg vt100.jpg

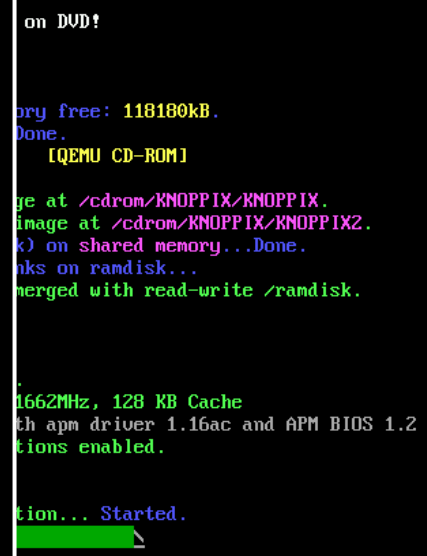
$ timg te



$ timg matrix.jpg





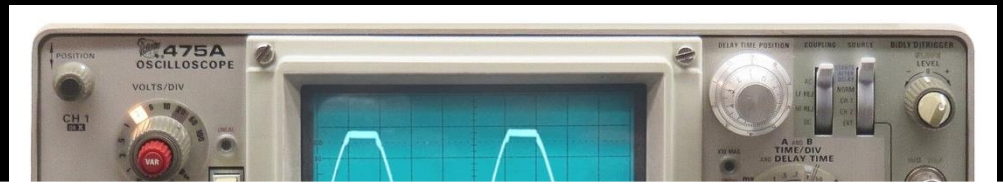


Dem
o01

Dem
o02

2.1 Il terminale del passato

Al tempo il terminale altro non era che lo schermo dove l'informazione veniva fornita all'operatore, non in forma di testo, ma in *forma* caso degli scherm





Quando il segnale è analogico e non necessita di essere ricordato, allora il ce. È *dar*: non data.

Il drastico calo dei costi di immagazzinare memoria e poi presentarla su schermo è stato uno dei motivi per cui gli schermi hanno sostituito la carta e preso il sopravvento. Un grazie va detto anche alle *televisioni*.



Il terminale dei tempi che furono risulta essere molto diverso dal concetto di terminale di oggi, e molto scomodo anche.

Dalla carta si è passati allo schermo, ma le funzionalità rimanevano le stesse dei terminali meccanici.

<u>Dumb terminal</u>

- Orientato ai solo caratteri
- Un carattere alla volta
- Una linea alla volta
- Limitate *escape sequences*
- Connessione con RS-232

Alcuni terminali permettevano input tramite *penna ottica*.

$ timg lightpen.jpg



/* Hello, world! in the C programming language */

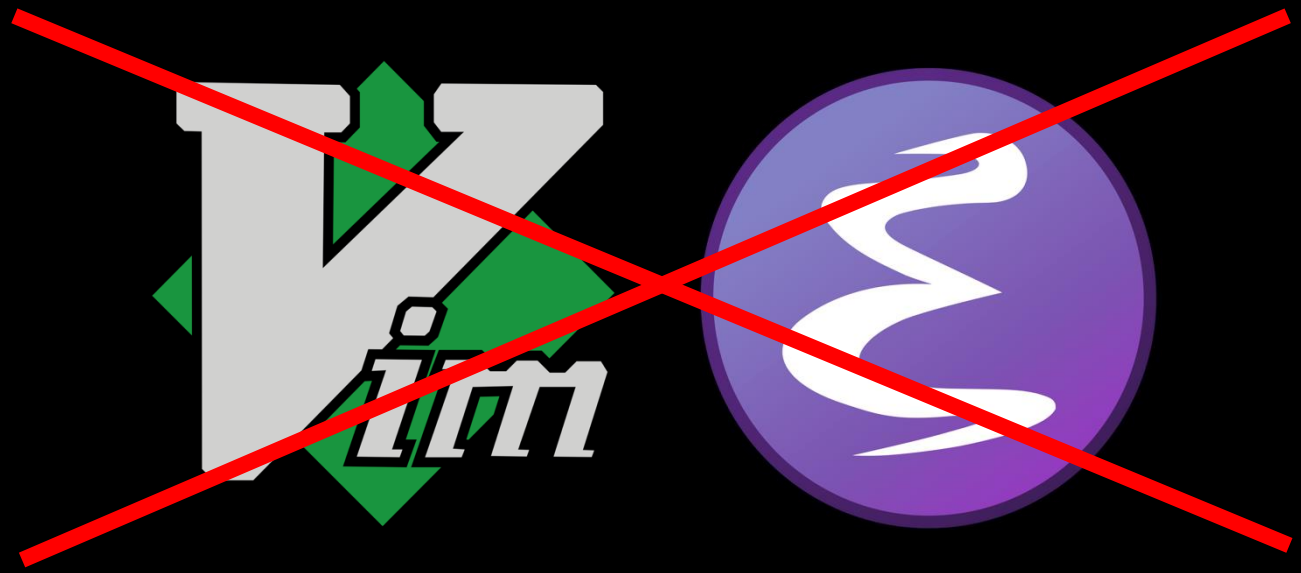
```
/* Hello, world! in the C programming language */

main()
```

```
/* Hello, world! in the C programming language */
main()
{
```

```
/* Hello, world! in the C programming language */
main()
{
pritnf("hello, world\n");
```

```
/* Hello, world! in the C programming language */
main()
{
pritnf("hello, world\n");
```



```
/* Hello, world! in the C programming language */
main()
{
pritnf("hello, world\n");
```

```
/* Hello, world! in the C programming language */
main()
{
pritnf("hello, world\n");
```

- «Enter»
- .
- «Enter»

```
/* Hello, world! in the C programming language */
main()
{
pritnf("hello, world\n");
.
```

- p4
- «Enter»

```
/* Hello, world! in the C programming language */
main()
{
pritnf("hello, world\n");
.
p4
pritnf("Hello World");
```

- s/pritnf/printf
- «Enter»

```
/* Hello, world! in the C programming language */
main()
{
pritnf("hello, world\n");
.
p4
pritnf("Hello, world");
s/pritnf/printf
```

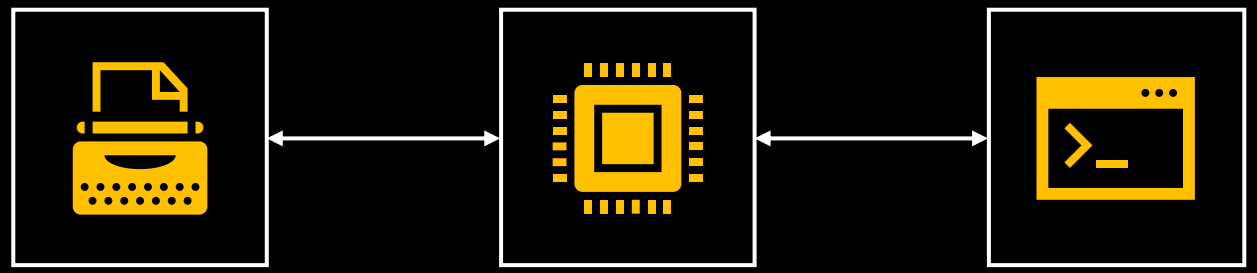
Dem
o03

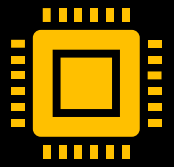
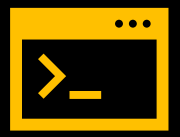
Una delle architetture di terminale passate le cui orme ancora oggi ripercorriamo, è quella implementata da UNIX V7, uno dei sistemi operativi più famosi della storia dell'informatica, creato da Dennis Ritchie e Ken Thompson, agli AT&T Laboratories. Scritto interamente in C, UNIX V7 include tutte le funzionalità più all'avanguardia dei sistemi che in quel tempo erano in esecuzione.



In UNIX, *tutto è un file*. Il terminale altro non è che un file nel sistema operativo, anche chiamato device driver. Più terminali possono esistere simultaneamente e il kernel ne fa accesso alla cartella /dev. Il sistema operativo siede tra la telescrivente e un programma in esecuzione chiamato shell.

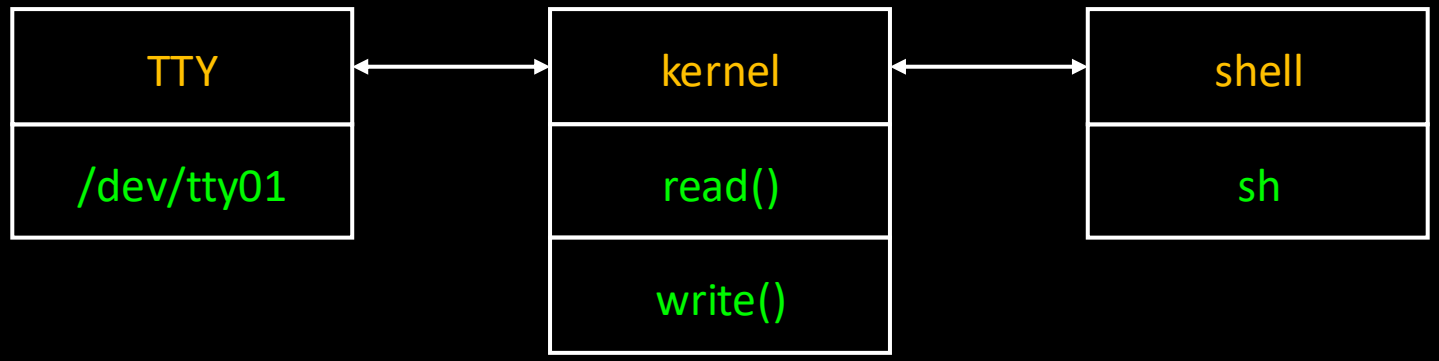
UNIX®
An Open Group Standard


TTY
kernel
shell

TTY		Invia un byte quando riceve un colpo e stampa su carta quando riceve un byte
kernel		Decide se l'input è valido oppure va scartato perché non nell'alfabeto. È inoltre capace di decidere se caratteri precedentemente inviati e salvati in memoria, vanno o meno scartati.
shell		Il programma in costante esecuzione che aspetta il completamento della linea. Esegue il comando presente nella linea se questo è valido.

Il programma di shell ci parla chiamando write() sul file /dev/tty01, viceversa legge i contenuti del file chiamando read(). Nel kernel UNIX, esistevano due tipi di operazioni chiamate *line discipline*.

- Cooked
  Questa è la modalità operativa di base. Il kernel siede tra la TTY e la shell come controllore: solo nel momento in cui una nuova linea viene inviata, allora la shell si risveglia.
- Raw
  Tutti gli input da tastiera vengono inviati alla shell direttamente. Questo sarà il modus operandi degli editor di testo che verranno.



2.2 Il terminale del presente

Il terminale del presente eredita molte delle funzionalità del terminale del passato.

- La modalità *cooked* è ancora presente e permette la visioni di caratteri di escape.
- I codici di escape ANSI usati negli anni 70 sono ancora gli stessi di oggi.

Perché cambiare qualcosa che già funziona?

Le differenze più eclatanti stanno nel come le informazioni vengono presentate, e quanto velocemente.

$ timg tty0.jpg

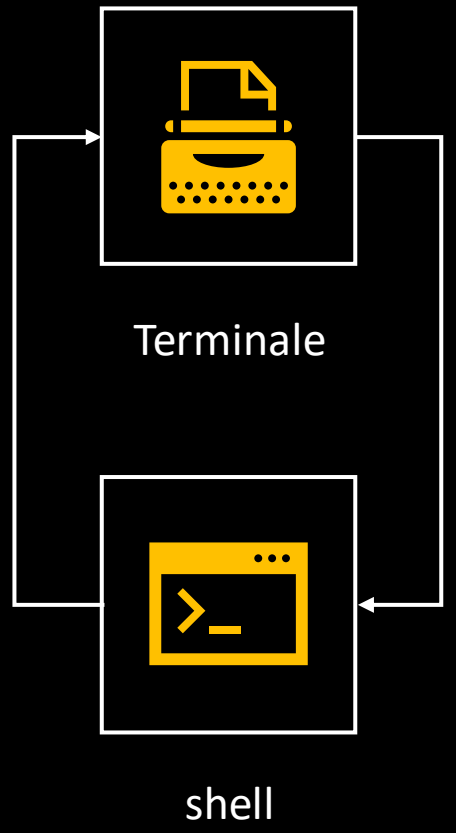


2.3 L’emulatore di terminale

Quello che oggi è comunemente chiamato «terminale» è in gergo tecnico chiamato emulatore di terminale, perché *emula* come un *terminale* era nei tempi passati (e in alcuni casi, ancora è).

Il kernel gestisce la conversazione tra l’emulatore di terminale e il sistema operativo con uno pseudo-terminale (PTY).

Una serie di tubi (o *pipes*) connettono l’emulatore di terminale con la shell.



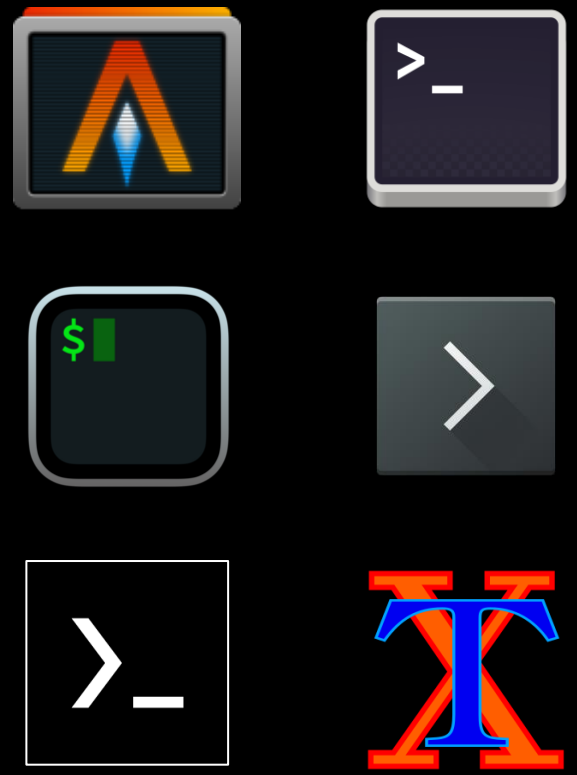
Dem
o04

3.0 Come accedere al terminale

Uno dei vantaggi e punti di forza dei sistemi basati su Unix, è che molto spesso questi rispettano *convenzioni*.

Il rispettare convenzioni permette a kernel come Linux di eseguire uno o più terminali simultaneamente, con uno o più emulatori di terminale simultaneamente.

Questo il motivo per cui esistono decine di emulatori di terminale, le cui funzionalità variano fortemente in base alle preferenze dei loro creatori, e degli utenti che lo usano.



Dem
o05

6. FAQ

- Perché conoscere il terminale è rilevante?
  Ci troviamo in un periodo storico dove i grandi magnati del software e hardware, dettano cosa possiamo e non possiamo fare con i dispositivi che noi abbiamo comprato, con i nostri soldi. Il software libero, l'ago della bilancia che permette l'infrastruttura moderna di funzionare, è minacciato. Alcune funzionalità prima facilmente accessibili, diventano col tempo bloccate con la scusa di «sicurezza», dietro pagamenti, dietro limitazioni, non allo scopo di «proteggere», ma allo scopo di «guadagnare». Il terminale permette l'accesso a impostazioni che in origine erano facilmente accessibili, ma che oggi non lo sono.

- Perché conoscere il terminale è utile?
  L'umanità si sta immergendo sempre di più nell'automazione. La nostra generazione, e le prossime che arriveranno, saranno sempre più dipendenti nei confronti delle macchine e conoscerle sarà la chiave per sbloccarne il pieno potenziale. Il terminale è uno degli strumenti che ci permette di fare ciò.

6. FAQ

- Devo sapere usare il terminale?
  Non ancora. Seppure vero che siamo in un’era digitale, non a tutti è tenuto sapere cosa significa «aprire un terminale» ed «eseguire comandi». È comunque uno strumento che nel migliore dei casi stampa a schermo dei messaggi umorosi, e nel peggiore elimina il disco con le foto di famiglia.

- Quando conviene usare un comando?
  Nel momento in cui notiamo che stiamo installando uno o due programmi di troppo per raggiungere un semplice scopo, come pulire il disco da file obsoleti, o liberare spazio in memoria. Un breve comando ha il potenziale di risolvere i nostri problemi.

- Devo sapere cosa fa un comando?
  Sì.

- Posso avere paura?
  Sì.

Fin.