Informatica e dintorni

di Emanuele Ferri

Revisione 1.2

Indice generale

Informatica

*L'**informatica** è la scienza che si occupa del trattamento dell'informazione mediante procedure automatizzate* (algoritmi), *avendo in particolare per oggetto lo studio dei fondamenti teorici dell'informazione, della sua computazione a livello logico e delle tecniche pratiche per la sua implementazione […]. Il termine italiano "informatica" deriva da quello francese "informatique", contrazione di informat(ion) (automat)ique [...]* ossia informazione automatica.

[https://it.wikipedia.org/wiki/Informatica]

Da notare che l'equivalente inglese di informatica è computer science, che pone enfasi sul concetto di computer, inteso come macchina in grado di eseguire algoritmi più che sugli algoritmi stessi. Ma che cos'è un algoritmo? Cos'è un computer?

Algoritmo

*La parola "**algoritmo**" è una combinazione tra la parola latina algorismus, che deriva dal nome di "Al-Khwarizmi", importante matematico arabo del nono secolo, e la parola greca arithmos, che significa "numero". Essa indica "l'arte di calcolare con facilità ed esattezza". Un algoritmo si può definire come un procedimento che consente di ottenere un risultato atteso eseguendo, in un determinato ordine, un insieme di passi semplici corrispondenti ad azioni scelte solitamente da un insieme finito.*

*[*http://www.iisgalilei.eu/areadiprogetto/2F2017/pagine_sito/informatica_algoritmo.html]

Ci sono due termini riportati in questa definizione su cui è il caso di soffermarsi. Il primo termine è "intuitivamente". Intuitivamente significa che questa definizione serve a capire il concetto ma non è rigorosa. Il secondo termine è "passi semplici". Ciò che per me è semplice potrebbe non esserlo per qualcun altro. Come esempi di algoritmi vengono talvolta riportate delle ricette di cucina. Vediamone una:

Chiffon cache

...

Preriscaldare il forno a 165 gradi.

Separare i tuorli dagli albumi.

Aprire la bacca di vaniglia ed estrarre i semini.

Passare lo zucchero per qualche secondo al frullatore in modo da renderlo piu' fine.

…

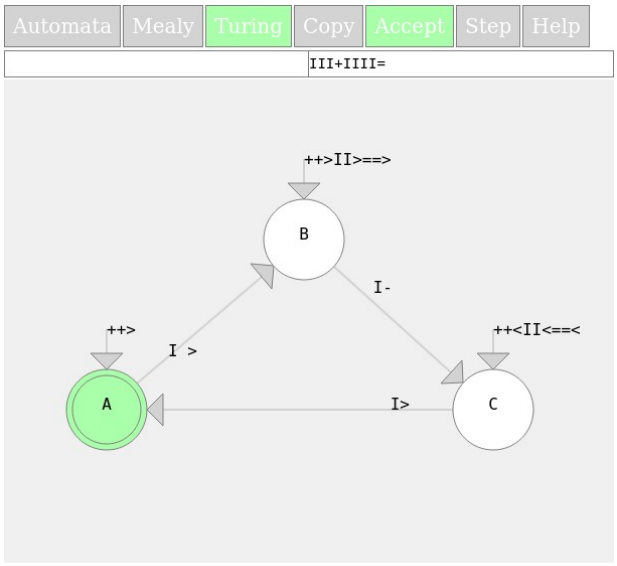
[http://www.kucinare.it/ricetta/Chiffon_cake-310.aspx]

Per qualcuno “separare i tuorli dagli albumi” può essere un “passo semplice”, atomico, inutile da spiegare ulteriormente. Altri potrebbero credere che le galline fanno uova sode.

Macchina di Turing

A tutte queste ambiguità è stato posto rimedio per la prima volta da Alan Turing. Turing inventò un modello di “macchine” immaginarie (che qualcuno con altrettanta immaginazione poi ha anche realizzato:https://www.youtube.com/watch?v=cYw2ewoO6c4). Il termine macchina non deve dunque trarre in inganno: non si tratta di dispositivi meccanici e/o elettronici ma di formalismi puramente concettuali. A differenza delle ricette di cucina dunque, queste “macchine” sono definibili in termini matematici rigorosi.

Si prenda per esempio la **macchina di Turing** (MdT) che esegue la somma tra numeri naturali rappresentati utilizzando gli stecchini, come farebbero dei bambini o degli uomini primitivi. La macchina è stata realizzata tramite il simulatore tursim. Graficamente la macchina ha questo aspetto:



I cerchi rappresentano gli stati, le frecce transizioni. Il cerchio verde è lo stato di partenza, da dove comincia il calcolo, quelli doppi, uno in questa macchina, rappresentano gli stati di accettazione: se la computazione termina in uno stato di accettazione è come se restituisse vero. Sulle transizioni ci sono delle etichette che contengono il simbolo letto, il simbolo scritto e lo spostamento della testina sul nastro separati da una virgola. Ci possono essere più transizioni che collegano la stessa coppia di stati. In questo simulatore le frecce si sovrappongono (le etichette però si affiancano una sull'altra per rendere possibile la lettura).

I passi elementari delle macchine di Turing sono fatti tutti allo stesso modo: se mi trovo in un certo stato e leggo un certo simbolo sul nastro, mi sposto in un altro stato, scrivo un simbolo su nastro e agisco sulla testina di lettura/scrittura spostandola a sinistra si uno (<), a destra di uno (>) o lasciandola dov'è (-).

In questa particolare macchina sommatrice i numeri vengono rappresentati mediante “stecchini” (il simbolo I), come farebbero dei bambini o degli uomini primitivi. Lo zero viene espresso mediante l'assenza di stecchini. Uno stecchino significa uno, due stecchini due etc. Il simbolo + tra i due numeri indica la somma, il simbolo =, che separa gli addendi dal risultato, rappresenta l'uguaglianza.

Le macchine di Turing forniscono un modello formale esente da interpretazioni soggettive. Le regole che le governano sono semplici per chiunque, indipendentemente dalla sua abilità culinaria. Turing andò oltre realizzando una macchina universale. Una **macchina di Turing universale** è una macchina di Turing capace di simulare qualunque altra macchina di Turing inserendo quest’ultima nel nastro insieme ai dati.

Negli anni i matematici e gli informatici teorici si sono posti la seguente domanda: siamo proprio sicuri che con le macchine di Turing si possano implementare tutti gli algoritmi o, in altre parole, siamo sicuri che la Macchina di Turing Universale possa eseguire tutti gli algoritmi? Purtroppo, proprio a causa dell'ambiguità della definizione di algoritmo non si può dare una risposta a questa domanda.

Negli anni che seguirono furono introdotti altri formalismi, come il lambda calcolo, le funzioni ricorsive, le macchine a registri illimitati, … . Talvolta è sembrato che alcuni di questi formalismi potessero risolvere problemi irrisolvibili dalle macchine di Turing. Contrariamente alle apparenze iniziali queste supposizioni si sono sempre dimostrate errate. Inoltre si è sempre riusciti a dimostrare che, nella migliore delle ipotesi, questi nuovi formalismi erano equivalenti alle MDT, tanto che si è coniato il termine **Turing-equivalente** per esprimere la potenza computazionale massima. Visti i continui fallimenti nella ricerca di qualcosa che andasse oltre le MDT, è stata anche formulata una ipotesi, detta **congettura di Church-Turing**: tale congettura afferma che le macchine di Turing possono calcolare tutto ciò che è calcolabile o, in altre parole, che le MdT sono un modo per descrivere gli algoritmi.

Programma

Un **programma** è la descrizione di un algoritmo mediante un linguaggio di programmazione. Quello che segue è un esempio di programma in Python, un

linguaggio di programmazione molto diffuso, che legge due numeri interi e stampa il massimo.

```
a=int(input('Inserisci il primo numero intero '))
b=int(input('Inserisci il secondo numero intero '))
print('Il massimo è')
if a>b:
  print(a)
else:
  print(b)
```

Esistono centinaia, forse migliaia di **linguaggi di programmazione** diversi ma, al di là delle loro differenze, la loro funzione è quella di implementare algoritmi. A differenza delle MdT i linguaggi di programmazione hanno molte più regole e necessitano di un certo tempo per essere padroneggiati discretamente, tuttavia, soprattutto con quelli più moderni e ad alto livello, risolvere problemi anche molto complessi è enormemente più facile rispetto alle MdT. Le MdT hanno una grande importanza nell'Informatica teorica, per esempio nelle dimostrazioni, mentre i linguaggi di programmazione vengono utilizzati per risolvere problemi concreti.

Computer

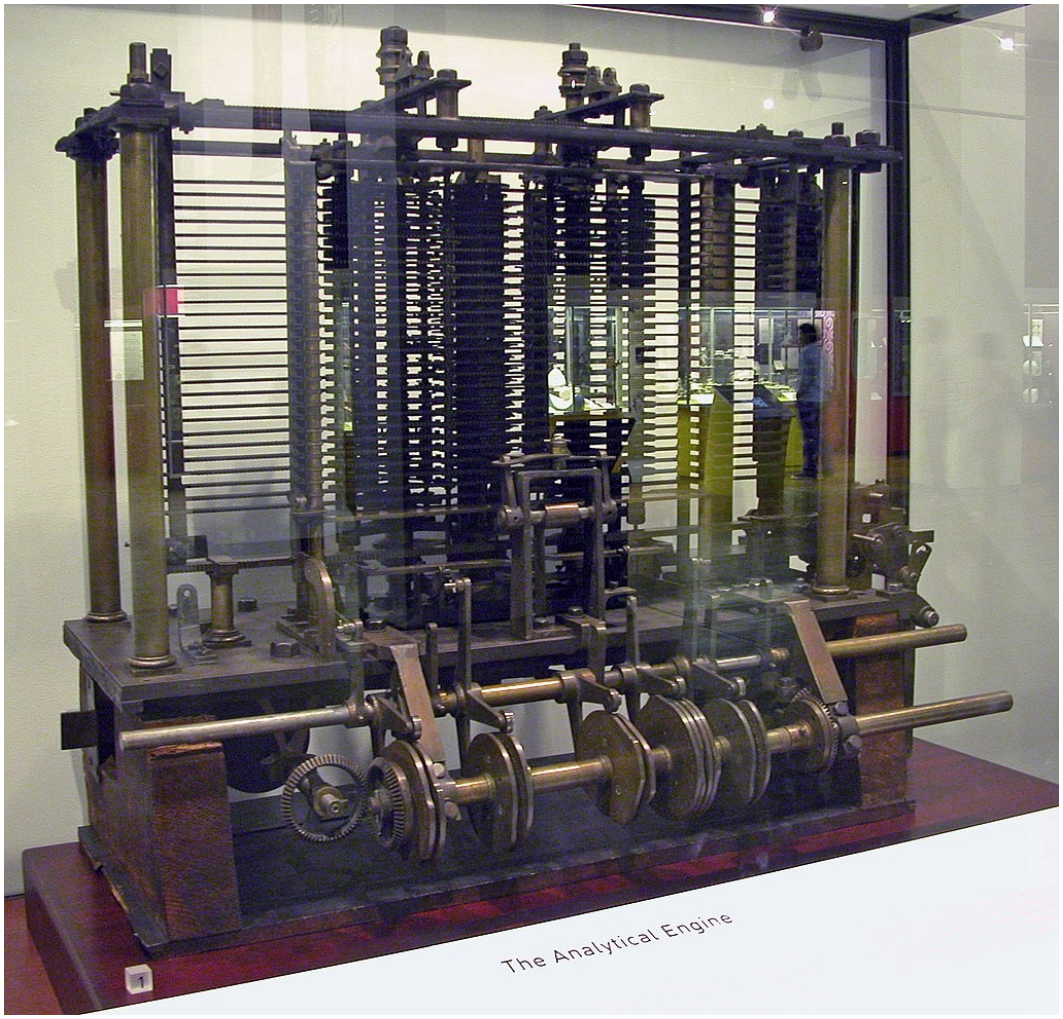
Un **computer** è una macchina programmabile in grado di eseguire qualunque algoritmo: è l'equivalente fisico di una macchina di Turing Universale.

A differenza di altri tipi di macchina, nei computer odierni le parti meccaniche sono poche o assenti e in genere non fondamentali. Più recentemente, il termine è stato impiegato, sia in ambito tecnico-scientifico che nel linguaggio corrente, per indicare oggetti costruiti dall'uomo con lo scopo di gestire non l'energia, ma l'informazione. Sia gli elaboratori elettronici in generale che alcune loro componenti, ed i loro modelli teorici sono quindi spesso indicati con tale termine. [https://it.wikipedia.org/wiki/Macchina]

Anche il Telaio Jacquard [https://it.wikipedia.org/wiki/Telaio_Jacquard] realizzato nel lontano 1801 era una macchina ed eseguiva dei programmi per disegnare sul tessuto ma non era un computer. Il motivo è che, oltre ad essere programmabile, su un computer si deve poter programmare qualunque algoritmo (nei limiti della sua memoria) mentre alcune macchine, seppur programmabili, possono eseguire solo alcuni tipi di programmi, a seconda dello scopo per cui sono state costruite.

Una definizione corretta di computer potrebbe essere “macchina programmabile Turing-equivalente” ossia macchina con la stessa capacità di calcolo delle macchine di Turing.

Secondo questa definizione il primo computer della storia è la Macchina Analitica [https://it.wikipedia.org/wiki/Macchina_analitica] di Charles Babbage [https://it.wikipedia.org/wiki/Charles_Babbage] che purtroppo non fu mai completata dal suo inventore a causa di problemi tecnici ed economici. Dal progetto che ci è pervenuto tuttavia si evince che quella macchina, se ultimata, sarebbe stata a tutti gli effetti un computer. Da notare che la Macchina Analitica era totalmente meccanica, il che confuta la tesi secondo cui i computer siano per forza dispositivi elettronici, per quanto la stragrande maggioranza di essi effettivamente lo sia. In pratica non è importante con che tecnologia è realizzato un computer per essere definito tale, l’importante è che esso possa eseguire qualunque algoritmo mediante la programmazione.



Modello di una parte dell'Analytical Engine di Babbage

Esempi di computer più diffusi oggi sono i personal computer desktop (fissi) e portatili, gli smartphone, le console da gioco.

Hardware e software

*Con il termine **hardware**, in Ingegneria Elettronica e Informatica, si indica la parte fisica di un computer, ovvero tutte quelle parti elettroniche, elettriche, meccaniche, magnetiche, ottiche che ne consentono il funzionamento.* [https://it.wikipedia.org/wiki/Hardware] Per essere utilizzabile l'hardware ha bisogno del software. Con il termine **software** si intendono i programmi. Per essere eseguito e archiviato il software necessita dell'hardware.

Il termine hardware in inglese significa “componente dura” in contrapposizione al software, la “componente morbida” che deve il suo nome agli albori dell’Informatica, durante la seconda guerra mondiale, quando le istruzioni della macchina Enigma venivano scritte su pagine solubili in acqua per facilitarne la distruzione ed evitare che cadessero in mani nemiche.

Analogico e digitale

*L'attributo **digitale** viene utilizzato in relazione ad una grandezza/sistema che varia fra un insieme finito di possibili stati in maniera discreta (ogni stato è ben distinto dagli altri). Deriva dal termine inglese digit (cifra). **Analogica** è, invece, una grandezza (ovvero un sistema) che varia con continuità (questo non significa che non si possa rappresentare numericamente). La parola deriva dal greco e significa, letteralmente, discorso simile.*

[http://www.provincia.bz.it/fp/gutenberg/approfondimenti/app10.html]

Un esempio di sistema digitale è il blocco del cambio della macchina: le marce in genere sono 6 più una posizione centrale chiamata "folle", non si può mettere la leva del cambio a metà tra la prima e la seconda. Il volante e l'acceleratore invece sono sistemi analogici in quanto non si muovono a scatti ma con continuità. Altri esempi di un controllo digitale lo troviamo nel phon: in genere il phon può essere spento, erogare una quantità moderata di aria o erogarne molta, quindi è dotato di una levetta che può assumere tre posizioni in coppia con un'altra simile per dosare il calore dell'aria. I tasti della tastiera del computer possono assumere solo due valori (premuto o rilasciato) quindi sono digitali. I testi del telecomando (compreso il volume) possono anch'essi assumere solo due valori mentre il volume dei vecchi televisori e dei vecchi impianti stereo venivano regolati tramite una manopola analogica. Fu la

Nintendo a creare, per il suo Nintendo 64, il primo controller dotato di "leva analogica", volgarmente detto "tricorno". Tuttavia nel tricorno il segnale viene digitalizzato ancor prima di raggiungere la console (c'è una specie di ruota dentata per ognuno dei due assi). Anche se Mario su N64 può correre, correre lentamente, camminare e stare fermo, non può certo replicare tutta la gamma di velocità di un essere umano reale. Parlando di mouse, anche se il movimento è analogico, il campionamento dello spostamento avviene in modo digitale, sia in quelli dotati di pallina che in quelli ad infrarossi. Sono analogici i rubinetti dell'acqua e colonnina di mercurio di un termometro, che sale e scende con continuità. Un dinamometro pesapersone, spesso volgarmente ed erroneamente chiamato bilancia, può mostrare il peso tramite una lancetta che si muove con continuità o avere un display che riporta un numero finito di valori. In entrambi i tipi di dinamomentro, tuttavia, la forza peso viene misurata tramite un modello analogico: la legge di Hooke. Un orologio a lancette non necessariamente è analogico: in molti orologi (tutti?) le lancette si muovono a scatti e per fare un giro completo scattano 60 volte. Le lampade di casa sono quasi sempre digitali (accese o spente) però ce ne sono alcune (tipo i faretti alogeni) la cui luminosità può essere dosata tramite una manopola o una leva dal movimento continuo.

Dopo aver fornito definizioni ed esempi che sembrano chiarire abbastanza bene le caratteristiche dei due sistemi, pongo la seguente domanda: "la clessidra è analogica o digitale"? Certamente, visto da lontano, il fluire della sabbia appare continuo, tuttavia sappiamo bene che avvicinandoci potremmo osservare la caduta dei singoli granelli, un fenomeno discreto. Potremmo utilizzare una sabbia più sottile o renderla sottilissima in modo artificiale ma ciò servirebbe solo ad ingannare un osservatore superficiale. Per quanti sforzi possiamo fare per renderla più sottile, questo processo ha un limite nella struttura atomica della materia. Forse alcuni fenomeni ci appaiono analogici perché la nostra osservazione della realtà è superficiale. Quello che conta però non è tanto la natura della realtà quanto piuttosto il modello, analogico o digitale, che utilizziamo per descriverla.

Quasi tutti i computer oggi sono digitali, tuttavia *esistono computer analogici, che utilizzano segnali analogici (come il voltaggio o la corrente) per rappresentare e manipolare dati. A differenza dei computer digitali, … , i computer analogici utilizzano circuiti elettronici per eseguire calcoli matematici con variabili continue.*

Un esempio di computer analogico è l'Analogue Computing Machine (ACM) utilizzato durante la seconda guerra mondiale per calcolare le traiettorie di missili. Altri esempi includono il Differential Analyzer, utilizzato per risolvere equazioni

differenziali, e il simulatore di volo analogico Link Trainer, utilizzato per addestrare i piloti.

Anche se i computer digitali hanno sostituito i computer analogici nella maggior parte delle applicazioni, i computer analogici sono ancora utilizzati in alcune aree specializzate, come l'analisi di segnali audio e la modellizzazione di sistemi fisici complessi.

Tratto da una conversazione con ChatGPT.

Numeri e sistemi di numerazione

Non è chiaramente il fine di questo testo dare una definizione rigorosa di numero. Mi preme tuttavia sottolineare un aspetto fondamentale: è importante non confondere un numero con il simbolo utilizzato per rappresentarlo. Per esempio, vi sono diversi modi per rappresentare il numero dodici:

- dodici lingua italiana;
- IIIIIIIIIII stecchini dei primitivi/tacche dei nemici uccisi sul calcio della pistola;
- XII numero romano;
- 12 notazione posizionale decimale;
- 1100 notazione posizionale binaria.

Sono tutti modi di rappresentare lo stesso numero, appartenenti a culture ed ambiti differenti. Tra loro però ce ne sono alcuni migliori di altri

Il **sistema di numerazione posizionale** è senza dubbio il migliore: ideato in India e migliorato dagli arabi, si è diffuso in tutto il mondo. In questo sistema di numerazione ogni cifra, in base alla sua posizione, ha un differente peso. Nel **sistema posizionale decimale** vengono usate dieci cifre (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) e per questo viene detto anche a base dieci. La prima cifra a partire da destra, dato che gli arabi scrivono da destra a sinistra, indica quante unità compongono il numero, la seconda quante decine, la terza quante centinaia, … .

Usando un sistema posizionale, tramite algoritmi che si basano solo sui simboli (cioè sul modo in cui sono scritti i numeri) si può facilmente:

- Rappresentare qualunque numero naturale. E' una operazione meno banale di quanto sembri: i romani, per esempio, non potevano farlo. Introducendo il simbolo meno (-) si possono scrivere tutti gli interi, anche quelli negativi, col simbolo di frazione (/) tutti i razionali, … .
- Confrontare due numeri naturali qualsiasi e dire qual è il maggiore. Confrontare due numeri significa confrontare i due insiemi aventi quel numero di elementi e vedere qual è più numeroso. Per esempio, per sapere se sono di più le penne o i tappi basta infilare le penne nei tappi e vedere cosa resta. Potrebbe essere un procedimento molto lungo già con numeri relativamente piccoli. Se rappresentiamo i numeri con la notazione posizionale, tutto si

riduce ad un confronto tra i simboli che rappresentano i due numeri. Il numero che ha più cifre è il maggiore. Se hanno lo stesso numero di cifre, partendo da quelle di peso maggiore, si controlla fino a che non se ne trovano due diverse: il numero con la cifra maggiore è il più grande. Se le cifre sono tutte uguali chiaramente i numeri sono uguali. Questo procedimento è banale e molto veloce anche per numeri grandissimi. Anche il sistema di numerazione romano permette di confrontare due numeri, ma non in modo così semplice.

- Addizionare due numeri naturali. Addizionare a e b significa contare il numero di elementi dell'unione di due insiemi disgiunti aventi rispettivamente a e b elementi. Detto in parole povere, 3+2 significa prendere un sacchetto con tre biglie, un altro con due, metterli inseme e contarle: è un procedimento che richiede molto tempo. L'algoritmo della somma che ci è stato insegnato, invece, è estremamente efficiente in quanto sfrutta il modo in cui sono scritti i numeri, cioè i simboli. Si sommano le unità e si tiene il resto, poi le decine con il resto, le centinaia col resto e così via. Per questo sommare due numeri anche molto grandi per noi è un gioco da ragazzi. Per gli antichi romani, anche operazioni semplici come la somma richiedevano un certo sforzo ed è per questo che, per calcolare, si aiutavano con dei sassolini. Calculus, in latino, significa proprio sassolino.
- ...

Chiaramente non esiste solo il solo sistema posizionale decimale. Dato che le basi sono infinite (due, tre, quattro, …, dieci, … sedici, ...) esistono infiniti sistemi di numerazione posizionale. La base dieci è quella che è stata adottata dagli esseri umani ma la più semplice per eseguire i calcoli è la base due ed è per questo che è quella comunemente adottata dai computer.

I primi sedici numeri naturali in base due:

```
0     0
1     1
2    10
3    11
4   100
5   101
6   110
7   111
8  1000
9  1001
```

```
10  1010
11  1011
12  1100
13  1101
14  1110
15  1111
```

Pensate per esempio alle regole apprese da bambini per sommare due numeri in base 10:

```
0+0=0
0+1=1
0+2=2
…
1+0=1
1+1=2
1+2=3
…
2+0=2
2+1=3
2+2=4
…
```

In tutto le combinazioni sono cento! Le regole per sommare numeri in base due invece sono solo quattro:

```
0+0=0
0+1=1
1+0=1
```
`1+1=10` ossia zero unità e una coppia.

I numeri scritti in binario sono generalmente molto più lunghi di quelli in base dieci ed è difficile determinarne la grandezza a colpo d’occhio, per questo non sono molto adatti agli esseri umani.

Cambiamenti di base

Se un computer adotta il sistema di numerazione binario, gli input vanno inseriti in base 2 e gli output vengono forniti anch'essi in questa base. Presumendo che gli utilizzatori siano degli esseri umani, si rendono pertanto necessarie due conversione: dalla base 10 alla base 2 per gli input, dalla base 2 alla base 10 per gli output. Queste

conversioni si potrebbero effettuare a mano, ma sarebbe un compito ingrato. In alternativa si potrebbero affidare alla macchina. Per i moderni computer, che adottano il sistema di numerazione binario, ovviamente esistono già programmi che implementano gli algoritmi di conversione. Le prime calcolatrici meccaniche, la macchina analitica di Babbage, l'IBM System/3, ... adottavano un sistema di numerazione decimale proprio per evitare di dover convertire gli input e gli output. Tuttavia questo approccio, se da un lato evita le conversioni, dall'altro complica notevolmente l'hardware.

Anche se esistono programmi appositi per passare da una base all'altra e nell'uso quotidiano del computer queste conversioni vengono fatte automaticamente, tuttavia è utile imparare come funzionano e soprattutto capire perché.

Conversione usando gli stecchini

Bisogna per prima cosa imparare a scrivere i numeri come i primitivi (non so se effettivamente i primitivi scrivessero i numeri così ma mi piace pensarlo). Ecco alcuni esempi:

0 (assenza di stecchini)

1 `I` (uno stecchino)

2 `II` (due stecchini)

3 `III` (tre stecchini)

…

Mi piace usare gli stecchini perché sono molto vicini all'essenza stessa del numero. Avrei potuto usare biglie, sassolini, cammelli, bisonti … ma gli stecchini sono più facili da disegnare.

Prendiamo ad esempio il numero decimale 27. Significa:

7 da `I`

2 da `IIIIIIIIII`

Ossia:

`IIIIIIIII IIIIIIIII I I I I I I I` (ho lasciato volutamente gli spazi)

Un procedimento analogo va applicato con la base due. Se vogliamo sapere che numero è 10011:

1 da `I`

1 da `II`

0 da `IIII`

0 da `IIIIIIII`

1 da `IIIIIIIIIIIIIIII`

Ossia

`IIIIIIIIIIIIIIII II I` (ho lasciato volutamente gli spazi)

Il bello della base due è la sua semplicità: i gruppi di stecchini o ci sono o non ci sono, non esistono vie di mezzo.

Questo procedimento ci permette di passare da una qualunque notazione posizionale agli stecchini (cioè all'essenza del numero). Il procedimento inverso, un po' più complicato ma non troppo, ci permette di passare dagli stecchini ad una qualunque notazione posizionale. Per esempio, proviamo ad esprimere il numero del primo esempio, che in decimale si scriveva 27 in forma binaria.

Potrei avere molti gruppi da `I`

`I I I I I I I I I I I I I I I I I I I I I I I I I I I`

Ma potrei fare anche meglio: 1 gruppo da `I` e molti gruppi da `II`

`II II II II II II II II II II II II II II I`

Meglio: 1 gruppo da `I`, 1 gruppo da `II` e molti gruppi da `IIII`

`IIII IIII IIII IIII IIII IIII II I`

Ancora meglio: 1 gruppo da `I`, 1 gruppo da `II` , 0 gruppi da `IIII` e molti gruppi da `IIIIIIII`

`IIIIIIII IIIIIIII IIIIIIII II I`

Per finire: 1 gruppo da `I`, 1 da `II`, 0 da `IIII`, 1 da `IIIIIIII`, 1 da `IIIIIIIIIIIIIIII`

`IIIIIIIIIIIIIIII IIIIIIII II I`

che nel sistema posizionale binario si scrive 11011

Ora proviamo a tradurre il numero del secondo esempio, che in binario si scriveva 10011 in notazione decimale.

Molti gruppi da I

I I I I I I I I I I I I I I I I I I

9 gruppi da I e 1 gruppo da IIIIIIIII

IIIIIIIII I I I I I I I I I

Che nel sistema posizionale decimale si scrive 19.

Notare che, per passare dalla notazione posizionale agli stecchini si effettuano delle moltiplicazioni (ogni cifra di un numero scritto in notazione posizionale indica quante volte prendere un gruppo di stecchini, quindi è una moltiplicazione) e poi si sommano i risultati (si uniscono gli stecchini). Al contrario, per passare dagli stecchini alla notazione posizionale, si effettuano delle divisioni successive (facendo i gruppi) e si tengono i resti.

Conversione senza stecchini

A questo punto è lecito chiedersi se sia necessario passare dagli stecchini per convertire un numero da una base a un'altra. La risposta è ovviamente no: gli stecchini (o le pedine) sono utili per comprendere come funziona l'algoritmo, ma **si può passare da una base all'altra scrivendo direttamente i numeri nel sistema posizionale**. Il problema è che siamo abituati fin dalle elementari a calcolare in base dieci ed effettuare conversioni dirette tra basi diverse dalla base dieci è un problema per la maggior parte di noi.

A titolo di esempio prendiamo il numero 19 e proviamo a convertirlo in base due senza passare dagli stecchini.

Il procedimento adottato per passare dagli stecchini alla base desiderata era quello di formare i gruppi (e poi gruppi di gruppi e così via) e tenere gli stecchini avanzati, ossia nell'eseguire delle divisioni successive tra interi tenendo i resti. Per scrivere il numero decimale 19 in base due basta dividerlo per due, dividere il risultato per due e così via fino ad ottenere 0 (e quindi non poterlo più dividere) tenendo i resti:

```
19/2=9  con resto di 1
9/2=4   con resto di 1
4/2=2   con resto di 0
2/2=1   con resto di 0
1/2=0   con resto di 1
```

I resti, partendo dall'ultimo, sono 10011, che è il numero cercato.

Potremmo adottare lo stesso metodo anche per eseguire l'operazione inversa, ossia **convertire 10011 da binario a decimale**, tuttavia, per chi come noi è abituato a fare i calcoli in base dieci, risulta più semplice sommare i prodotti dei "pesi" (le potenze della base) per le cifre e sommarli. Per esempio

```
16       8        4        2        1
1        0        0        1        1
16x1+    8x0+     4x0+     2x1+     1x1=     19
```

Vediamo un altro esempio di conversione, questa volta da base dieci a base tre. Il numero da convertire è 35.

```
35
11   2
3    2
1    0
0    1
```

I resti, partendo dall'ultimo, sono 1022, che è il numero cercato in base due.

Ora convertiamo il numero 1022 da base tre a base dieci. In questo caso applicare l'algoritmo dei resti risulta difficile a noi che siamo abituati a calcolare in base dieci, pertanto ci conviene sommare i prodotti dei "pesi" per le cifre.

```
27       9        3        1
1        0        2        2
27x1+    9x0+     3x2+     1x2=     35
```

Considerazioni "avanzate"

Ci è stato insegnato fin dalle elementari ad eseguire calcoli in base dieci e quindi per noi è difficile farli usando altre basi, ma ciò non significa che sia più difficile in assoluto. Eseguire calcoli in base due, per esempio, è decisamente più semplice ed è per questo che in genere è la base utilizzata dai computer.

Proviamo per esempio a convertire il numero binario 10011 da base due a base dieci (1010) operando divisioni successive e tenendo i resti.

```
10011:1010=1 con resto di 1001 che in decimale è 9
1:1010=0 con resto di 1010 che in decimale è 1
```

Il numero in decimale si scrive 19. Per funzionare funziona, tuttavia dobbiamo già sapere come si scrivono in binario tutti i numeri da zero a dieci (nel caso specifico uno, nove ed ovviamente dieci) e saper fare i calcoli in base due.

Vediamo ora come convertire 19 da base dieci a base due sommando i prodotti dei pesi per le cifre. In binario 1 si scrive 1, 9 si scrive 1001 e 10 si scrive 1010. 19 significa una decina e nove unità che tradotto in binario significa 1 da 1010 e 1001 da 1.

```
1010     1
1        1001
1010x1+  1x1001=  10011
```

Funziona, basta saper sommare e moltiplicare in base due.

Questo metodo ha un difetto: per applicarlo devo già sapere come scrivere in binario tutti i numeri compresi tra zero e dieci (nel caso specifico mi servono uno, nove e dieci).

A questo punto viene spontaneo chiedersi quale dei due metodi convenga adottare per effettuare una conversione da una base all'altra. La risposta è: se la base di partenza è maggiore conviene eseguire le divisioni successive (perché i resti saranno sempre cifre presenti in entrambe le basi), se invece la base di partenza è minore conviene sommare tra loro le cifre del numero moltiplicate per il corrispettivo peso (perché le cifre che compongono il numero saranno sempre presenti in entrambe le basi).

Il sistema di numerazione esadecimale

Come avrete ormai capito, la base due sarà anche adatta ai computer ma non lo è altrettando per gli esseri umani. Per esempio, anche numeri relativamente piccoli sono composti da molte cifre e non trasmettono le reali dimensioni del numero con una semplice occhiata, come accade per i numeri scritti in base dieci. Per ovviare a questo problema nell'Informatica si ricorre spesso alla base sedici, per esempio per i colori delle pagine web o per rappresentare indirizzi di memoria in linguaggio assembly. La base sedici è ancora più "compatta" della base dieci e semplifica molto le conversioni con la base due.

Il sistema numerico esadecimale (spesso abbreviato come esa o hex) è un sistema numerico posizionale in base 16, cioè che utilizza 16 simboli invece dei 10 del sistema numerico decimale tradizionale. Per l'esadecimale si usano in genere i simboli da 0 a 9 per le prime dieci cifre e poi le lettere da A a F per le successive sei cifre, per un totale di 16 simboli.

[https://it.wikipedia.org/wiki/Sistema_numerico_esadecimale]

La tabella che segue mostra gli stessi numeri, da zero a quindici, scritti in decimale, esadecimale e binario.

Dieci	Sedici	Due
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Come si può notare, per scrivere un numero binario fino a quattro cifre è sufficiente una sola cifra esadecimale. Per numeri binari composti da cinque fino a otto ne servono due e così via. Per esempio, per convertire il numero binario 10001010 è sufficiente convertire il primo gruppo da quattro e poi le il secondo gruppo: $1000 \rightarrow 8$ e $1010 \rightarrow$ A quindi in esadecimale il numero si scrive 8A.

Il numero esadecimale 63 si converte in $6 \rightarrow 0110$, $3 \rightarrow 0011$ e in questo caso vanno aggiunti degli zeri davanti al numero per ottenere 01100011 ossia 1100011.

Unità di misura dell’informazione

Bit

Il **bit**, dall’inglese **binary digit** (**cifra binaria**) è l’unità di base dell'informazione. Un bit può assumere solo due valori che siamo soliti chiamare **0** e **1**, anche se falso e vero, no e sì, nero e bianco, croce e testa, spento e acceso, male e bene, yin e yang, ... andrebbero altrettanto bene. Il simbolo del bit è la lettera **b** minuscola.

Oltre al bit, esistono altre unità di informazione, per esempio il trit, dall’inglese ternary numeral system (sistema numerico ternario) che può assumere tre valori differenti. Per quanto siano effettivamente esistiti computer che adottavano unità di informazione diverse dal bit, per esempio basate su tre o dieci valori, **praticamente tutti i computer oggi utilizzano una logica binaria** e le memorie informatiche generalmente utilizzano i bit per memorizzare i dati.

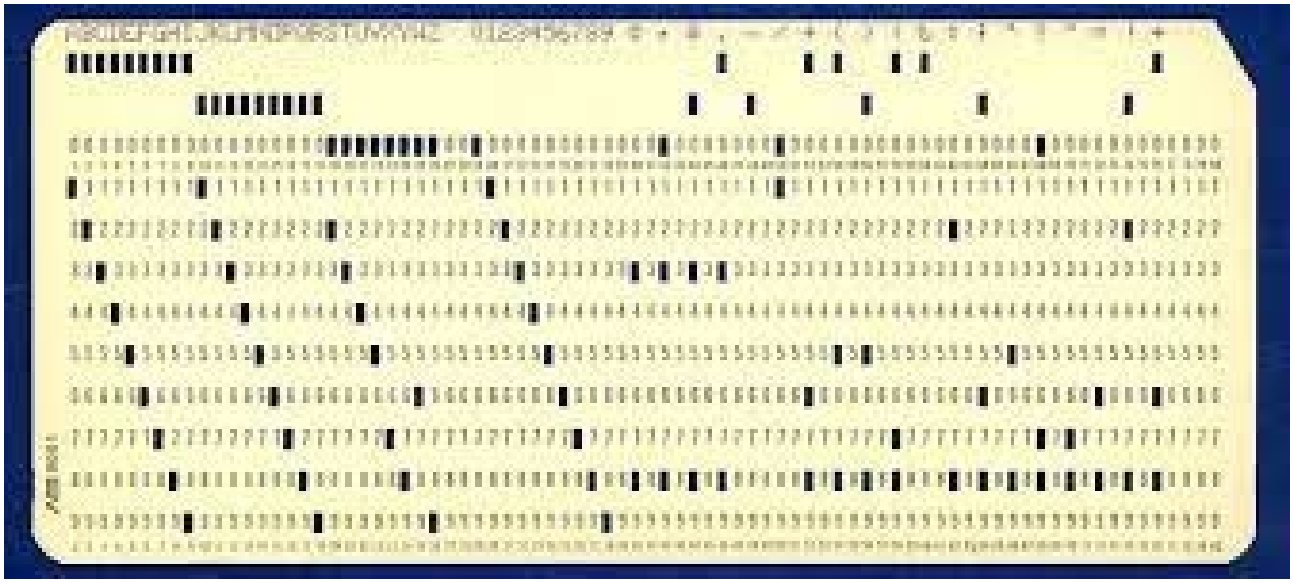


Figura 1: Una scheda perforata di cartone. I bit possono valere zero (assenza di foro) oppure uno (foro presente).

Byte

Un **byte** è una sequenza, un insieme ordinato di bit. Sono esistiti byte composti da diverse quantità di bit, tuttavia, dal 1964, la dimensione del byte viene definita di otto bit. Il simbolo del byte è la lettera **B** maiuscola. Byte deriva dalla parola inglese bite, che significa boccone, morso ed è la più piccola porzione di memoria a cui si può accedere in lettura o scrittura, ossia che può essere “addentata”.

Volendo rappresentare schematicamente un byte verrebbe fuori una cosa del genere:

croce	croce	testa	testa	croce	croce	croce	testa

Non ho usato 0 e 1 volutamente, altrimenti qualcuno potrebbe pensare che i byte contengono solo dati numerici. Un byte potrebbe rappresentare il codice operativo di una istruzione di un programma, un carattere, un numero o altro. Se vogliamo conformarci al sistema possiamo rappresentare un byte in questo modo:

0	0	1	1	0	0	0	1

Il contenuto di questo byte non ha alcun significato particolare, è stato messo a caso.

Un byte può assumere 256 possibili valori che vanno da 00000000 a 11111111. Un bit può assumere due valori. Due bit possono assumere quattro valori. Ogni bit aggiunto moltiplica per due le combinazioni ottenibili coi bit precedenti e siccome in un byte i bit sono otto, il numero di combinazioni possibili è $2x2x2x2x2x2x2x2=2^8=256$. 256 valori diversi sono più che sufficienti per rappresentare un carattere, parecchie istruzioni diverse, abbastanza tonalità di grigio affinché, passando da una all'altra, la sfumatura, seppur discreta, appaia continua, …

In sostanza, anche se l'unità di base dell'informazione è il bit, molto spesso, anche se non sempre, si preferisce utilizzare il byte.

Rappresentazione dei numeri

I computer rappresentano i numeri interi positivi (naturali) in modo analogo a come vengono scritti nella notazione posizionale binaria, ma gli interi negativi, oltre a un simbolo per lo zero (0) e per l'uno (1), necessitano anche di un terzo simbolo, il meno (-). Per scrivere gli interi negativi pertanto sono state adottate diverse rappresentazioni. Una di queste, detta "in segno e modulo", consiste nel riservare un bit al segno: 0 significa +, 1 significa -. Di solito però i computer non adottano questo sistema ma preferiscono rappresentare i numeri negativi in "complemento a due", eliminando la necessità di un hardware dedicato alla sottrazione.

I numeri razionali in matematica vengono scritti usando il simbolo di frazione oppure la virgola. Il problema di usare la virgola è che può portare a infinite cifre periodiche. Nei computer sono state adottate due rappresentazioni dei numeri razionali: con virgola fissa e con virgola mobile. La virgola mobile è preferibile ed è il sistema più comune.

Anche se talvolta in Informatica si fa riferimento ai numeri con la virgola con il termine “real”, non esiste alcun modo per un computer di rappresentare i reali irrazionali che vengono approssimati a dei semplici razionali.

Rappresentazione dei caratteri

Il metodo più diffuso per rappresentare i caratteri, stampabili e non, usando un solo byte è la codifica **ASCII** (American Standard Code for Information Interchange). I primi 128 caratterisono uguali per tutti, gli altri 128 cambiano a seconda della lingua.

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Prefissi per multipli decimali

Chiaramente spesso si ha la necessità di trattare con quantità molto grandi e per questo motivo vengono utilizzati i prefissi del sistema internazionale di unità di misura, gli stessi che si usano in fisica, in bicicletta, dal panettiere, … . I prefissi utilizzati sono solo quelli ad esponente positivo, non avendo senso suddividere un byte.

kilo: $k=10^3$

mega: $M=10^6$

giga: $G=10^9$

tera: $T=10^{12}$

peta: $P=10^{15}$

Attenzione che maiuscole e minuscole fanno una enorme differenza. Per esempio m sta per milli, ossia un millesimo, mentre M sta per mega, ossia un milione. I prefissi maiuscoli indicano esponenti positivi, i minuscoli negativi, tranne purtroppo il

prefisso di kilo che è k (minuscolo) in quanto K (maiuscolo) significa kelvin (unità di misura della temperatura).

Se vogliamo misurare grossi quantitativi si byte:

kilobyte: $kB=10^3B$

megabyte: $MB=10^6B$

gigabyte: $GB=10^9B$

terabyte: $TB=10^{12}B$

petabyte: $PB=10^{15}B$

Prefissi per multipli binari

Generalmente le memorie informatiche hanno una capacità equivalente a una potenza di due, piuttosto che una potenza di dieci. Il motivo sarò chiarito più avanti. Accade spesso, tuttavia, che quando si fa riferimento a grossi quantitativi di memoria si approssimi la loro capacità alla potenza di dieci più vicina per sfruttare i prefissi del SI. Per esempio, quando si parla di un kB di RAM in genere ci si riferisce a 1.024 byte, quando si parla un MB a 1.048.576 byte, quando si parla di un GB a 1.073.741.824 byte. Molti programmi e molti libri di testo adottano questa convenzione errata. Per ovviare al problema il SI ha ideato i prefissi per multipli binari: kibi (ki), mebi (Mi), gibi (Gi), tebi (Ti), pebi (Pi), ...

kibi: $Ki=2^{10}\sim10^3=k$

mebi: $Mi=2^{20}\sim10^6=M$

gibi: $Gi=2^{30}\sim10^9=G$

tebi: $Ti=2^{40}\sim10^{12}=T$

pebi: $Pi=2^{50}\sim10^{15}=P$

(~ significa all'incirca uguale a) e quindi ovviamente:

kibibyte: $KiB=2^{10}\sim10^3=kB$

mebibyte: $MiB=2^{20}\sim10^6=MB$

gibibyte: $GiB=2^{30}\sim10^9=GB$

tebibyte: $TiB=2^{40}\sim10^{12}=TB$

pebibyte: $PiB=2^{50}\sim10^{15}=PB$

Classificazione della memoria

Le memoria informatiche si possono rappresentare come una sequenza di bit.. Qualunque sia il tipo di memoria utilizzato, usare solo due possibili valori rende più facile la lettura e la scrittura e gli errori sono meno frequenti.

Una piccola memoria di cinque byte la possiamo immaginare così

0	0	1	1	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	0	1	0	1	1	1	0
0	0	0	0	1	1	1	1

I valori dei bit nella tabella sono messi a caso, non significano niente.

In ambito informatico la **memoria** è la parte del computer destinata a conservare informazioni per un certo periodo di tempo. La memorizzazione di informazioni in memoria, e il successivo recupero delle medesime, sono funzioni fondamentali nel funzionamento del computer.

Non tutte le memorie sono digitali. Musicassette e videocassette generalmente venivano utilizzate in analogico, anche se sono esistiti supporti digitali per la musica e i video. Le stesse cassette usate in analogico per la musica erano ampiamente utilizzate come supporto digitale per i programmi dei vecchi home computer come il Commodore 64, lo ZX Spectrum, gli MSX e molti altri.

Nelle memorie analogiche, la copia non è mai identica all'originale: se copiate una musicassetta, per esempio con un semplice mangiacassette a doppia piastra, la differenza tra la copia e l'originale è percepibile all'ascolto. Al contrario, una copia digitale è perfettamente identica all'originale: questo è uno dei motivi del graduale abbandono della memorizzazione analogica a favore di quella digitale. Oggi le memorie analogiche talvolta vengono ancora usate, nel cinema per esempio, ma la maggior parte delle memorie sono digitali e binarie. L'utilizzo di memorie digitali ha portato a una sostanziale convergenza dell'informatica e del mondo dell'audio/video.

Sulla pagina di Wikipedia https://it.wikipedia.org/wiki/Memoria_(informatica) troviamo una classificazione della memoria molto dettagliata.

Viene spesso dato per scontato che la memoria di un computer sia un dispositivo elettronico o magnetico. In genere è vero, però bisogna fare le dovute precisazioni. Le schede perforate, che hanno conosciuto una larga diffusione anche in epoche relativamente recenti (fino ai primi anni '80), sono memorie informatiche a tutti gli effetti eppure non sono elettroniche né magnetiche: sono pezzi di cartone con dei buchi. Questo discorso serve a ribadire il concetto che, se da un lato è assodato che l'informatica deve il suo enorme successo all'evoluzione dell'elettronica, d'altro, a livello teorico, non è molto importante il modo con cui vengono implementati certi meccanismi hardware.

Modalità di accesso

Ad un dispositivo di memorizzazione si può accedere sostanzialmente in tre modi diversi.

Un esempio classico di memorie ad **accesso sequenziale** sono i nastri magnetici, come per esempio le musicassette. Per leggere o scrivere una locazione che mi interessa devo scorrere il nastro, come quando accedo a un canale premendo il tasto “canale successivo”. Un modo per trovare una canzone è ascoltare ogni tanto premendo play e, in base ad un indice delle canzoni (su carta o nella mia testa) andare avanti o indietro coi tasti fast-forward e rewind. Un altro celebre esempio di accesso sequenziale è il nastro delle macchine di Turing in cui la testina si muove sul nastro solo di una posizione alla volta e per capire dove si trova legge il nastro.

Nelle memorie ad **accesso diretto** invece si può "saltare" da una locazione di memoria all'altra fornendo un indirizzo. Nelle memorie ad accesso diretto il tempo per accedere ai dati non è costante ma dipende dall'indirizzo di memoria a cui è avvenuto l'accesso precedente. In ambito informatico l'esempio più comune di memoria ad accesso diretto è l'hard disk. Negli hard disk non è possibile accedere ai singoli byte: la minima porzione di disco leggibile e scrivibile è chiamata settore e tipicamente contiene 512 byte. Ogni settore ha un indirizzo. Il tempo di accesso al settore dipende dalla posizione corrente delle testine rispetto al settore: se il settore è vicino il tempo di accesso è basso, se è lontano alto. Ciò è dovuto alla natura in parte meccanica degli hard disk. I piatti girano, tutti insieme, velocissimi, e le testine si spostano avanti e indietro, collegate allo stesso braccio, vicinissime alla superficie dei dischi. Questi spostamenti richiedono più tempo se la strada da percorrere è lunga, meno se è breve.

Anche nelle memoria ad **accesso casuale**, dette **RAM** (**Random Access Memory**) si accede a una locazione di memoria in base all'indirizzo fornito ma, a differenza delle

memorie ad accesso diretto, il tempo di accesso è costante, come quando digito il canale sul telecomando del televisore.

Nelle RAM si ha la possibilità di indirizzare ogni singolo byte. Indirizzare è un po' come chiamare un giocatore col numero della sua maglia.

0	0	0	1	1	0	0	0	1
1	0	1	0	0	1	0	1	0
2	0	1	0	1	0	0	1	0
3	0	0	1	0	1	1	1	0
4	0	0	0	0	1	1	1	1

Nella prima colonna di sinistra, in grassetto, sono riportati gli indirizzi dei byte che,si possono scrivere anche in forma binaria:

0	0	0	1	1	0	0	0	1
1	0	1	0	0	1	0	1	0
10	0	1	0	1	0	0	1	0
11	0	0	1	0	1	1	1	0
100	0	0	0	0	1	1	1	1

Gli indirizzi dei byte sono dei numeri e pertanto hanno un ordinamento (10 viene prima di 100), si possono sommare (10+1=11) sottrarre etc.

Possibilità di scrittura

La tipica memoria può essere sia letta che scritta. Questi dispositivi sono detti **memorie a lettura-scrittura**. Esempi sono le memorie DDR, EPROM, EEPROM, dischi ottici CD-RW, dischi ottici DVD-RW, memorie elettroniche flash, nuclei di ferrite, dischi rigidi, floppy disk, dischi magneto-ottici RW.

Vi sono poi memorie che vengono scritte solo in fase di inizializzazione, e per le quali non è possibile la scrittura nell'uso normale. Tale inizializzazione può essere effettuata in modo incrementale dalla stessa apparecchiatura con cui vengono riletti i dati scritti. Questi dispositivi sono detti **memorie scrivibili una sola volta**, o

WORM (**Write Once, Read Many**). Esempi sono i dischi magneto-ottici WORM, dischi ottici CD-R, dischi ottici DVD-R, memorie elettroniche PROM, memorie elettroniche OTPROM.

Alternativamente, può essere necessario scrivere tutti i dati con un'apposita apparecchiatura esterna prima di poter usare la memoria in lettura. Tali memorie non sono più scrivibili e vengono dette memorie a sola lettura, o **ROM** (**Read-Only Memory**). Esempi: memorie elettroniche ROM, dischi ottici CD-ROM, dischi ottici DVD-ROM. … .

Volatilità e permanenza

Una memoria si definisce **volatile** se deve essere costantemente alimentata per mantenere il proprio contenuto. Un memoria si definisce **permanente** se mantiene il proprio contenuto anche in assenza di alimentazione.

Tecnologia costruttiva

Come già accennato le memorie possono essere fatte di materiali differenti.

Memoria cartacea

- Scheda perforata
- Nastro perforato

Memoria elettronica

- DRAM
- SRAM
- Memoria a stato solido

Memoria magnetica

- Memoria a nucleo magnetico
- Nastro magnetico (Musicassetta, VHS, ...)
- Disco magnetico (Floppy disk, Hard disk, Zip drive)

Memoria ottica

- Disco ottico (CD, DVD, Blue-ray Disc)

Memoria olografica

Da notare che le musicassette e le VHS non sono nate come memorie informatiche bensì per contenere rispettivamente musica e video in formato analogico, tuttavia sono state usate anche per archiviare dati in digitale.

Velocità di accesso e costo unitario

La velocità di una memoria si misura in **byte al secondo** (**B/s**). Per le memorie a lettura-scrittura, il tempo necessario per leggere un byte è normalmente vicino al tempo richiesto per scriverlo, per cui si parla genericamente di **tempo di accesso**. Per alcune memorie, per esempio quelle scrivibili una volta sola, la scrittura può essere molto più lenta della lettura; in tal caso, dato che la memoria verrà letta molte volte, si fa coincidere il tempo di accesso col tempo di lettura.

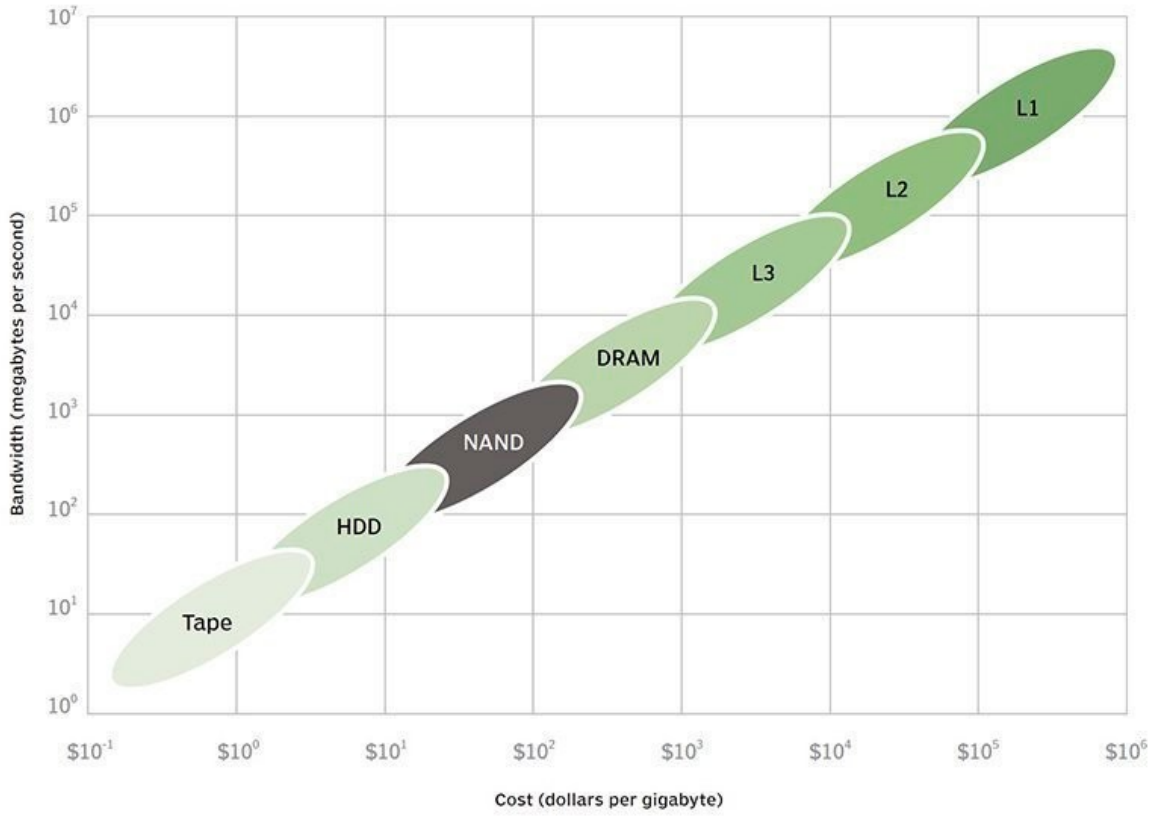
Nelle memorie ad accesso diretto e nelle memorie ad accesso sequenziale, il tempo di accesso è influenzato dal **tempo di posizionamento** delle testine. Negli hard disk, per esempio, il tempo di posizionamento coincide nel peggiore dei casi col tempo necessario per effettuare un giro completo, mediamente col tempo richiesto perché il disco faccia mezzo giro e nullo nel caso molto fortunato in cui la testina si trovi già posizionata sopra il settore richiesto. Nei nastri non è possibile stimare il tempo di posizionamento che potrebbe essere altissimo se i dati richiesti si trovano molto lontani dalla testina.

In generale, il costo unitario (cioè per byte) delle memorie cresce al crescere della velocità di lettura. Pertanto, la classificazione per velocità di lettura coincide sostanzialmente con la classificazione per costo unitario. Nell’immagine che segue, sull’asse delle ascisse troviamo il costo per GB mentre sull’asse delle ordinate la velocità espressa in MB/s. Da notare che al crescere del costo cresce proporzionalmente anche la velocità e che entrambi gli assi adottano una scala logaritmica.

FIGURE 1

The memory-storage hierarchy

Each orb represents a layer of memory or storage by speed and cost.



SOURCE: OBJECTIVE ANALYSIS, 2017

Hardware

La maggioranza dei moderni computer deriva dal modello ideato da Von Neumann per realizzare alcuni dei primi calcolatori elettronici. A questo schema di progettazione viene solitamente attribuito il nome di **modello (o architettura) di Von Neumann**. Esistono altri modelli alternativi, come per esempio il modello Harvard.

Memoria centrale

I computer sono dotati di una memoria digitale, binaria e ad accesso casuale che prende il nome di **memoria centrale**, o anche memoria principale. Nel modello di Von Neumann, in questa memoria si trovano i programmi da eseguire e i dati. Da notare che esistono anche architetture, come per esempio l'architettura Harvard, in cui programmi e dati occupano memorie fisicamente distinte. Il fatto che programmi e dati condividano la stessa memoria fisica permette di assegnare più o meno spazio agli uni o agli altri a seconda delle necessità.

Essendo una memoria ad accesso casuale, si ha la possibilità di indirizzare ogni singolo byte. Indirizzare è un po' come chiamare un giocatore col numero della sua maglia.

0	0	0	1	1	0	0	0	1
1	0	1	0	0	1	0	1	0
2	0	1	0	1	0	0	1	0
3	0	0	1	0	1	1	1	0
4	0	0	0	0	1	1	1	1

Nella prima colonna di sinistra, in grassetto, sono riportati gli indirizzi dei byte che si possono scrivere più propriamente in binario.

0	0	0	1	1	0	0	0	1
1	0	1	0	0	1	0	1	0
10	0	1	0	1	0	0	1	0
11	0	0	1	0	1	1	1	0

100	0	0	0	0	1	1	1	1

La memoria centrale deve essere estremamente veloce e quindi, purtroppo, è volatile. In tempi recenti viene realizzata con tecnologia DRAM. Veniva affiancata da una piccola ROM, ovviamente non volatile, in cui risiedevano, tra le altre, le routine di codice necessarie per l'avvio del computer. In tempi più recenti si è cominciato ad usare memorie all'occorrenza scrivibili.

Processore centrale

Il **processore centrale**, detto semplicemente processore o anche **CPU** (dall'inglese Central Processing Unit), è quel dispositivo hardware adibito all'esecuzione delle istruzioni di un programma. Possiede un **set (insieme) di istruzioni** hardware che è in grado di eseguire e che varia in base al tipo di processore. Quello che non è in grado di fare a livello hardware grazie al suo set di istruzioni viene implementato a livello software, mediante la programmazione.

Da notare che nel modello di Von Neumann non compare il termine processore. Il processore è una invenzione posteriore e integra al suo interno tre componenti che verranno discusse in seguito, ossia l'unità aritmetico logica, i registri e l'unità di controllo.

I diversi tipi di processore possono differire molto tra loro, per il modo in cui sono costruiti e per le istruzioni che sono in grado di eseguire. Le istruzioni che il processore non è in grado di eseguire nativamente si possono implementare mediante la programmazione partendo da quelle disponibili. Supponiamo per esempio di avere una processore capace di fare solo la somma tra numeri interi: con un semplice programma è facile ottenere anche la moltiplicazione e persino l'elevamento a potenza, a differenza di una calcolatrice, che può eseguire solo le operazioni per cui è stata progettata. Il problema insomma non è tanto cosa un computer è in grado di fare ma quanto in fretta riesce a farlo (memoria permettendo).

Istruzione

Un'**istruzione** di un programma è una sequenza di byte che dice al processore quale operazione compiere e con quali valori. Per esempio, si può immaginare che una istruzione fatta in questo modo

```
10101010
00000001
```

```
00000010
00000111
```

stia dicendo al processore di sommare (10101010) il numero 1 (00000001) al numero 2 (00000010) e di mettere il risultato nell'indirizzo di memoria 7 (00000111).

E' importante tenere presente che questo è solo un esempio inventato per dare una idea di massima del significato di istruzione. E' un esempio semplice da capire ma per certi versi abbastanza anomalo per via della scrittura in memoria finale, che di solito viene eseguita da un'altra istruzione. Il fatto che il primo byte significhi proprio somma è una scelta del tutto arbitraria. Il primo byte dell'esempio è detto **codice operativo**, gli altri tre prendono il nome di **operandi**. Tra gli operandi, i primi due sono costanti, il terzo è l'indirizzo di una locazione di memoria il cui contenuto può cambiare (una variabile). Ci sono anche istruzioni che fanno riferimento a sole locazioni di memoria. Per esempio, nella nostra architettura immaginaria ci potrebbe anche essere una istruzione che somma il contenuto di due locazioni di memoria e mette il risultato in una terza locazione. Nei processori reali, tuttavia, in genere una istruzione contiene al massimo un riferimento alla memoria.

Unità aritmetico-logica

ALU sta per **arithmetic logic unit** (unità aritmetico logica). Si può immaginare la ALU come una calcolatrice un po' particolare in grado di eseguire, oltre ad alcune operazioni aritmetiche, anche operazioni logiche e di confronto. Mentre le operazioni aritmetiche operano su numeri e restituiscono numeri, quelle logiche operano su valori di verità (vero e falso) e restituiscono valori di verità. Le operazioni di confronto operano su numeri e restituiscono valori di verità. È la ALU che si occupa materialmente di eseguire le istruzioni e quindi è da essa che dipende il set di istruzioni del processore.

Registri

I registri sono delle piccole memorie interne alla CPU. Quando viene eseguita un'istruzione, gli operandi vengono caricati in registri e il risultato del calcolo viene anch'esso inserito in un registro. I calcoli dunque non vengono eseguiti operando direttamente su locazioni di memoria centrale e neppure il risultato è inserito immediatamente in una locazione di memoria ma sempre attingendo a registri e scrivendo in registri.

Generalmente una architettura è detta ad n bit quando è di n bit la larghezza standard di una variabile semplice, per esempio (ma non solo) un numero intero.

Generalmente questo si riflette sulla dimensione dei registri della CPU, anche se non tutti i registri hanno necessariamente quella dimensione.

Utilizzare interi da 8 bit significa avere a disposizione 2^8=256 valori, da 0 a 255 se non si usa segno, da -128 a 127 se si utilizza il segno. Se fosse necessario operare su numeri più grandi è sempre possibile farlo in più passaggi, a discapito però della velocità del calcolo. Utilizzare interi a 16 bit invece significa disporre di 2^{16}= 65536 valori, da 0 a 65535 se si rinuncia al segno, da -32768 a 32767 se si utilizza il segno. Anche in questo caso, se fosse necessario operare su numeri di dimensioni maggiori è sempre possibile farlo in più passaggi.

Diversi modelli di processore possono avere un numero diverso di registri di diversa grandezza ma con funzioni simili. Vi sono alcuni registri presenti, con piccole differenze, in quasi tutte le architetture, come per esempio il program counter, che contiene l'indirizzo corrente del programma in esecuzione, l'instruction register, che contiene il codice operativo dell'istruzione corrente, il memory address register che indica l'indirizzo della locazione di memoria a cui si vuole accedere, … .

Unità di controllo

L'unità di controllo è una parte del processore adibita all'esecuzione del **ciclo macchina**, ossia dei seguenti compiti a rotazione:

1. Fetch: Consiste nel prelevamento dalla memoria del codice operativo dell'istruzione. Il codice operativo viene caricato nell'instruction register).
2. Decode: Consiste nella decodifica del codice operativo. Per esempio, se l'istruzione è una somma, la ALU viene impostata per eseguire una somma, se è un salto ad un indirizzo assoluto, la ALU non viene proprio usata, … .
3. Operand assembly: Consiste nel prelevamento degli operandi. Per esempio, se l'istruzione è una somma tra due costanti, vengono prelevate dalla memoria le due costanti, se la somma ha tra i suoi operandi una locazione di memoria, prima si preleva l'indirizzo della locazione, poi il contenuto. Notare come questa fase possa variare molto in relazione al codice operativo dell'istruzione. La maggioranza delle architetture moderne sono dotate di registri generali in cui spesso vengono caricati gli operandi (da sommare, sottrarre, …) e scritto il risultato dell'operazione svolta (nel procesore 8086, l'antesignano dei moderni processori per pc, questi registri si chiamano AX, BX, CX, ...)

4. Execute: Consiste nell'esecuzione dell'istruzione. Se l'operazione è un salto verrà effettuato il salto, se è una somma verrà eseguita la somma, … . Il risultato dell'operazione finisce anch'esso in un registro.
5. Store: Consiste nella scrittura del risultato nella memoria centrale. In questa fase il contenuto di un registro (in cui magari, nella fase di execute, era finito il risultato di una operazione aritmetica) viene copiato in una locazione della memoria centrale. La fase di store non è presente in tutte le istruzioni. Il risultato potrebbe benissimo restare in un registro, pronto per essere usato da una istruzione successiva, senza essere scritto nella memoria centrale, in modo da limitare gli accessi alla memoria.

Terminato un'istruzione, si passa all'istruzione successiva.

Dispositivi di input/output

I dispositivi di input/output servono a scambiare dati con l'esterno. I dispositivi di input li ricevono, quelli di output li inviano, quelli di input/output entrambe le cose. Alcuni dispositivi servono ad interfacciare i computer con gli esseri umani, altri ad interfacciare i computer con altri computer, altri ancora a memorizzare dati in modo permanente, magari per essere trasferiti su altri computer in un secondo tempo. Al tempo di Von Neumann non esisteva nessuno dei moderni dispositivi di input/output e quelli esistenti servivano principalmente ad interfacciare la macchina con gli esseri umani.

Alcuni esempi di dispositivi odierni piuttosto comuni:

• Da umano a computer (input): Tastiera, mouse, gamepad, scanner, microfono;

• Da computer ad umano (output): Monitor, stampante, casse acustiche;

• Da computer ad umano e viceversa (input/output): Touch screen, stampante multifunzione;

• Da computer a computer e viceversa (input/output): scheda di rete;

• Da memoria di massa a computer (input): lettore CD e DVD;

• Da computer a memoria di massa e viceversa (input/output): masterizzatore CD e DVD, hard disk (sia rimovibile che interno), chiavetta USB, registratore di nastri magnetici.

A differenza della memoria centrale e del processore, nessun dispositivo di I/O è fondamentale per il funzionamento del computer. In alcuni dispositivi le tastiere

vengono sostituite da touch-screen, gli hard disk da unità allo stato solido. Le casse acustiche sono comode ma se ne potrebbe fare a meno. Fino a non molti anni fa le schede per reti cablate erano presenti di rado mentre nei computer moderni sono state affiancate e in certi casi sostituite da dispositivi wireless. Nei computer degli anni ottanta (primi anni novanta) gli hard disk spesso non erano presenti (i miei primi computer utilizzavano musicassette come memorie di massa). Normalmente i siti web vengono ospitati da computer dotati solo di scheda di rete, privi di monitor, mouse e tastiera.

I dispositivi di I/O sono molti e cambiano nel tempo. Nessuno può dirsi indispensabile ma chiaramente ne serve almeno uno di input ed uno di output (o uno di input/output). In caso contrario il computer non potrebbe comunicare con l'esterno.

Tastiera

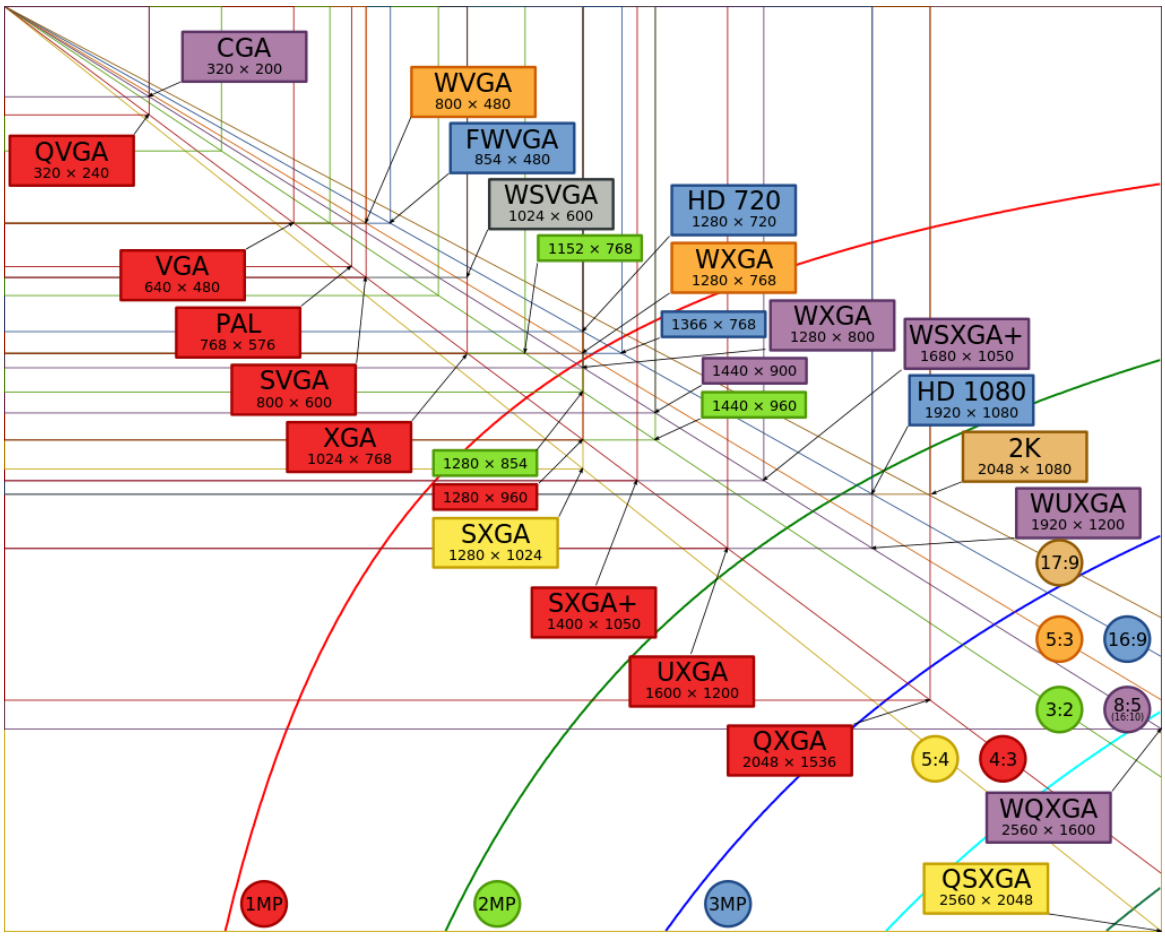
Mouse

Memorie di massa

Il software e i documenti come di testo, le immagini, i video, … vengono salvati nelle **memorie di massa**. Esempi di memorie di massa sono gli hard disk, i drive allo stato solido, le chiavette usb, i dischi ottici, … Le memorie di massa sono a tutti gli effetti dispositivi di I/O.

Schermo

Lo **schermo** è il dispositivo su cui vengono visualizzate le immagini e nei dispositivi moderni è formato una griglia di **pixel**. Un pixel è come una minuscola lampadina che può cambiare colore. Può appartenere a un monitor o a un televisore ed essere realizzato mediante diverse tecnologie (LCD, OLED, ...). Nel caso dei proiettori, lo “schermo di proiezione” è il telo o la parete su cui si proietta. La **risoluzione dello schermo** è il numero di pixel orizzontali e verticali che compongono lo schermo. Alcune risoluzioni molto comuni in passato erano 640×480, 800×600, 1024×768, mentre le risoluzioni più utilizzate oggi sono 1920×1080, 2560×1440, 3840×2160. C’è molta confusione coi nomi delle risoluzioni. L’immagine che segue è molto esaustiva anche se per motivi di spazio non arriva a 3840×2160, la risoluzione comunemente detta “4K” o più correttamente “Ultra HD”. Le risoluzioni più comuni in passato avevano un rapporto d’aspetto (il rapporto tra la risoluzione orizzontale e quella verticale) di 4/3 mentre quelle attuali generalmente sono in 16/9.



Di Original uploader was XXV at en.wikipediaLater version(s) were uploaded by Jjalocha, Aihtdikh at en.wikipedia. - Transferred from en.wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4022444

La **risoluzione grafica** è il numero di pixel orizzontali e verticali di una immagine. La scheda grafica produce l'immagine complessiva visualizzata sullo schermo, detto **frame** (fotogramma). Se la risoluzione del frame è inferiore a quella dello schermo, l'immagine va ingrandita (**upscaling**), se è maggiore, va rimpicciolita (**downscaling**). In linea generale, la risoluzione del frame dovrebbe essere uguale a quella dello schermo. Se il frame ha una risoluzione inferiore a quella dello schermo l'immagine apparirà "slavata". Se il frame ha una risoluzione superiore a quella dello schermo la scheda grafica sarà sottoposta a un carico di lavoro maggiore sostanzialmente inutile, se si esclude un certo "ammorbidimento" dei contorni ottenibile con molto meno sforzo mediante l'antialising.

Il numero di frame prodotti dalla scheda grafica in un secondo viene detto **framerate**. Maggiore è il framerate, più fluida è l'esperienza d'uso. Per esempio, in un utilizzo desktop, un framerate basso rende lo spostamento del mouse e delle finestre scattoso. Nei videogiochi un framerate basso peggiora l'esperienza di gioco e le performance del giocatore.

La **frequenza di refresh** è una proprietà dello schermo che indica quante volte al secondo questo si ridisegna. In genere gli schermi hanno una frequenza di refresh di 60Hz ma oggi non è raro trovare schermi con frequenza di refresh di 75Hz, 120Hz, 144Hz o addirittura più alte.

In genere si cerca di adottare un framerate “sincronizzato” (possibilmente uguale) alla frequenza di refresh dello schermo. Quando il carico di lavoro della scheda grafica è molto alto, come per esempio in certi giochi, può accadere che non sia possibile garantire un framerate stabile, per questo alcuni schermi sono dotati di frequenza di refresh variabile per sincronizzarla al framerate (tecnologia g-sync o freesync). In ogni caso, un framerate basso incide negativamente sull’esperienza d’uso.

In certi giochi, per evitare di abbassare il framerate rendendo il gioco “scattoso” si preferisce abbassare dinamicamente la risoluzione mantenendo il framerate stabile: si sacrifica un po’ la qualità grafica nelle situazioni più concitate per mantenere intatta la giocabilità.

Bus

I bus sono dei canali di comunicazione che collegano processore, memoria centrale e dispositivi di I/O. Limitiamoci alla comunicazione tra processore e memoria centrale, anche perché ai tempi di Von Neumann i dispositivi di I/O erano quasi assenti. In origine i bus erano realizzati mediante fili elettrici in rame e, anche se nei computer reali moderni la loro struttura è un po' differente e piuttosto complicata, concettualmente le cose non sono cambiate molto.

Vi sono tre bus:

Bus dati

Il bus dati serve a trasportare dati dalla memoria centrale al processore e viceversa. In altre parole questo bus dice COSA viene letto o scritto.

Bus indirizzi

Il bus indirizzi serve a selezionare la locazione di memoria in cui si devono effettuare le operazioni di lettura e scrittura. In altre parole dice DOVE leggere o scrivere.

La larghezza del bus indirizzi, ossia al numero di “fili” di cui si compone il bus, determina il numero totale di indirizzi e di conseguenza i byte in memoria indirizzabili. Poiché ogni “filo” può assumere due valori (tensione bassa o tensione alta, 0 o 1), il numero di combinazioni possibili è 2^n, dove n è il numero di “fili”.

Per esempio con un “filo” abbiamo solamente 2^1=2sole combinazioni: 0 e 1.

Con due 2^2=4: 00, 01, 10, 11.

Con tre 2^3=8: 000, 001, ..., 110, 111.

Con quattro 2^4=16: 0000, 0001, ..., 1101, 1111.

…

Con trentadue 2^{32}=4294967296: da 0000000000000000 a 1111111111111111 che è proprio il limite di indirizzamento sistemi a 32 bit, uno dei motivi per l’introduzione delle architetture a 64 bit (tramite PAE è possibile superare questo limite, ma non per la singola applicazione).

...

Con 64 ne abbiamo 2^{64}=18446744073709551616: da 0000000000000000000000000000000000000000000000000000000000000000 a 1111111111111111111111111111111111111111111111111111111111111111, un numero enorme di locazioni di memoria.

...

Poiché i “fili” del bus indirizzi sono un numero finito ed invariabile che dipende dall’architettura, risulta chiaro che la quantità massima di indirizzi è finita ed è sempre una potenza di due, motivo per cui sono stati introdotti i prefissi per multipli binari per misurare la capacità delle memorie. Da notare che questo limite c’è nelle memorie ad accesso diretto e casuale, non nelle memorie ad accesso sequenziale.

Bus di controllo

Il **bus di controllo** serve, tra le altre cose, ad indicare se va effettuata una operazione di lettura, una operazione di scrittura o nessuna delle due. In altre parole dice SE leggere, scrivere o non effettuare nessuna delle due operazioni.

Software

Classificazione

In genere si è soliti distinguere il software in:

Software di sistema

Il **software di sistema** è un tipo di software che fornisce ai computer le funzioni di base necessarie per gestire l'hardware e gli altri software, inclusi i driver del dispositivo, i programmi di utilità, i programmi di gestione delle risorse e i sistemi operativi.

Software di base

Fanno parte dei software di base quei programmi di utilità generale impiegabili da altri programmi, come le librerie software.

Software applicativo

Il software applicativo comprende qualunque programma utilizzabile dall'utente per svolgere particolari attività, come per esempio la scrittura, l'elaborazione di immagini, la gestione dei dati e altro. I software di office automation, detti anche software di produttività personale, fanno parte della categoria dei software applicativi. I più diffusi software di office automation sono LibreOffice, Microsoft Office e anche la suite di programmi di Google.

Licenze

Una licenza informatica (o licenza d'uso), in informatica, è il contratto con il quale il titolare dei diritti di sfruttamento economico sul software (programma informatico) definisce il regime giuridico di circolazione e le limitazioni nell'utilizzo e nella cessione dell'opera (che sia un'opera creativa, o un software, inteso come programma). [https://it.wikipedia.org/wiki/Licenza_(informatica)]

Software freeware

Il termine freeware indica un software che viene distribuito in modo gratuito.

Il freeware è distribuito indifferentemente con o senza codice sorgente, a totale discrezione dell'autore e senza alcun obbligo al riguardo. È sottoposto esplicitamente

ad una licenza che ne permette la redistribuzione gratuita. Il software freeware viene concesso in uso senza alcun corrispettivo, ed è liberamente duplicabile e distribuibile, con pochissime eccezioni.

Di norma l'autore che decide di rilasciare il suo lavoro come freeware, esercitando appieno il suo diritto di scegliere le forme e le modalità di distribuzione che ritiene più idonee, inserisce esplicitamente delle clausole che impediscono qualsiasi tipo di pagamento per la distribuzione del suo software, fatto salvo un eventuale "piccolo" rimborso per supporti e spese di duplicazione, esattamente come avviene per lo shareware.

Altri esempi di restrizioni opzionalmente inserite dagli autori sono le limitazioni all'utilizzo da parte di istituzioni statali o forze armate. (Tratto da http://it.wikipedia.org/wiki/Freeware)

Esempi di software freeware sono il videogioco Hurrican, il gestore di archivi iZarc, il lettore di PDF PDF-XChange Viewer, l' antivirus Antivir Personal Edition. Su tali programmi c'è da fare alcune considerazioni: i primi due software sono gratuiti per chiunque ed in ogni loro forma, il terzo ha una versione che si limita a visualizzare i PDF che è gratuita per tutti, un'altra, in grado di crearli e modificarli che invece è a pagamento, mentre Antivir è gratuito nella sua versione personal edition che però è limitata ai soli privati e quindi non può essere usata dalle aziende.

Software shareware

Lo shareware è una tipologia di licenza software molto popolare sin dai primi anni novanta. Vengono distribuiti sotto tale licenza in genere programmi facilmente scaricabili via Internet o contenuti in CD e DVD quasi sempre allegati alle riviste di Informatica in vendita in edicola.

Il software sotto tale licenza può essere liberamente ridistribuito, e può essere utilizzato per un periodo di tempo di prova variabile (generalmente 30 giorni). Scaduti questi termini, per continuare ad utilizzare il software è necessario registrarlo presso la casa produttrice, pagandone l'importo. All'avvio dell'applicazione shareware generalmente un nag screen informa l'utente su come effettuare la registrazione e sulle condizioni di utilizzo.

La versione di prova può avere, in aggiunta o in alternativa alla durata limitata, rispetto alla versione completa, limitazioni quali l'impossibilità di stampare o salvare i file o simili, numero di utilizzi limitato, contenere al suo interno meccanismi di protezione tali da impedire di utilizzare il software dopo la scadenza, mancanza di

supporto del produttore, watermarks audio o video sovraimposti ai file multimediali prodotti e altro... (Tratto da http://it.wikipedia.org/wiki/Shareware)

Un classico esempio esempio di software distribuiti tramite questo tipo di licenza sono le demo giocabili dei videogiochi. In pratica si tratta di versioni perfettamente giocabili ma contenenti solo alcuni livelli. Il sofware per masterizzare Alcohol 120% viene ditribuito in una versione di prova della durata di 15 giorni.

Software libero (free software)

L'espressione free software (software libero) si riferisce alla libertà dell'utente di eseguire, copiare, distribuire, studiare, cambiare e migliorare il software. Più precisamente, esso si riferisce a quattro tipi di libertà per gli utenti del software:

0. Libertà di eseguire il programma come si desidera, per qualsiasi scopo.
1. Libertà di studiare come funziona il programma e di modificarlo in modo da adattarlo alle proprie necessità. L'accesso al codice sorgente ne è un prerequisito.
2. Libertà di ridistribuire copie in modo da aiutare il prossimo.
3. Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti da voi apportati (e le vostre versioni modificate in genere), in modo tale che tutta la comunità ne tragga beneficio. L'accesso al codice sorgente ne è un prerequisito.

[https://www.gnu.org/philosophy/free-sw.it.html]

Esistono numerose licenze di software libero: la più famosa è probabilmente la GNU GPL di Stallman.

Le licenze che garantiscono queste libertà sono licenze di software libero. E' da notare che non si accenna al prezzo. Il software libero infatti può essere venduto da chiunque a qualunque prezzo. Ovviamente un altro distributore può fare altrettanto e il prezzo spesso è poco superiore al costo di distribuzione ma ibero non implica necessariamente gratis (anche se spesso è così)!!!!!

Software copyleft

Il concetto di **copyleft** è in netta contrapposizione a quello di copyright. Una programma è copyleft [https://it.wikipedia.org/wiki/Copyleft] se è libero ma possiede anche una importante restrizione: non è permesso cambiare licenza, anche in seguito

a pesanti modifiche. In genere non è permesso integrare software copyleft a software commerciale e distribuirlo. La GNU GPL è una licenza fortemente copyleft.

N.B. Non tutto il software libero è copyleft!!!

Free software non protetto da copyleft

Se nella licenza di un software libero non vengono espressamente indicate restrizioni per chi lo ridistribuisce, allora non è considerata copyleft.

Software di pubblico dominio (con sorgenti di pubblico dominio)

Anche i sorgenti devono essere di pubblico dominio altrimenti non è free software. Il software di dominio pubblico è privo di copyright. Questo tipo di software è libero ma non avendo un copyright chiunque può mettergli il proprio e associarli una qualsiasi licenza. Questo implica che alcune copie, o varianti di questo software possono non essere più libere.

Software Open Source

Software **open source** ha un significato simile a software libero (free) ma visto da una prospettiva diversa. Spesso vengono considerati sinonimi, anche se in realtà esistono delle differenze, soprattutto da un punto di vista "filosofico". La definizione di open source è più tecnica e meno ideologica. Le licenze open source in genere sono meno restrittive (per esempio non sono copyleft come la GPL) e piacciono di più alle imprese.

Quelli che seguono 10 punti che contraddistinguono il software open source (non occorre saperli, basta avere una idea generale):

1) Libera ridistribuzione. La licenza non può limitare nessuna delle parti nella vendita o nella fornitura di software come componente di una distribuzione di software aggregati, contenente programmi provenienti da fonti diverse. La licenza non può richiedere il pagamento di una royalty o di diritti per tale rivendita.
2) Codice sorgente. Il programma deve includere il codice sorgente, e deve consentire la distribuzione sia sotto forma di codice sorgente sia in forma compilata. Nei casi in cui un prodotto non venga distribuito con il codice sorgente, deve esserci la possibilità, ben pubblicata, di scaricare il codice sorgente via Internet senza costi aggiuntivi. Il codice sorgente deve essere la forma privilegiata in cui in programmatore modificherà il programma. Codice

sorgente deliberatamente nascosto non è ammesso. Forme mediate, come l'output di un preprocessore non sono ammesse.

3) Prodotti derivati. La licenza deve consentire l'attuazione di modifiche e di prodotti derivati, consentendo inoltre la loro distribuzione sotto gli stessi termini di licenza del software originale.

4) Integrità del codice sorgente dell'autore. La licenza può imporre limitazioni sulla distribuzione del codice sorgente in forma modificata solamente se la licenza consente la distribuzione di file "patch" insieme al codice sorgente con lo scopo di modificare il programma durante l'esecuzione del build. La licenza deve consentire esplicitamente la distribuzione di software realizzato a partire dal codice sorgente modificato. La licenza può richiedere che i prodotti derivati portino un nome o un numero di versione diverso dal software originale.

5) Nessuna discriminazione verso singoli o gruppi. La licenza non deve porre discriminazioni verso qualsiasi persona o gruppo di persone.

6) Nessuna discriminazione verso campi di applicazione. La licenza non deve porre limitazioni sull'uso del programma in un particolare campo di applicazione. Per esempio, non può impedire l'uso del programma in una azienda o per la ricerca genetica.

7) Distribuzione della licenza. I diritti allegati al programma devono applicarsi a tutti coloro a cui viene ridistribuito il programma, senza la necessità di applicare una licenza supplementare per queste parti.

8) La licenza non deve essere specifica per un prodotto. I diritti allegati al programma non devono dipendere dal fatto che il programma faccia parte di una distribuzione particolare. Se il programma viene estratto da tale distribuzione e usato o distribuito nei termini della licenza del programma, tutte le parti a cui il programma viene ridistribuito devono avere gli stessi diritti garantiti in occasione della distribuzione originale del software.

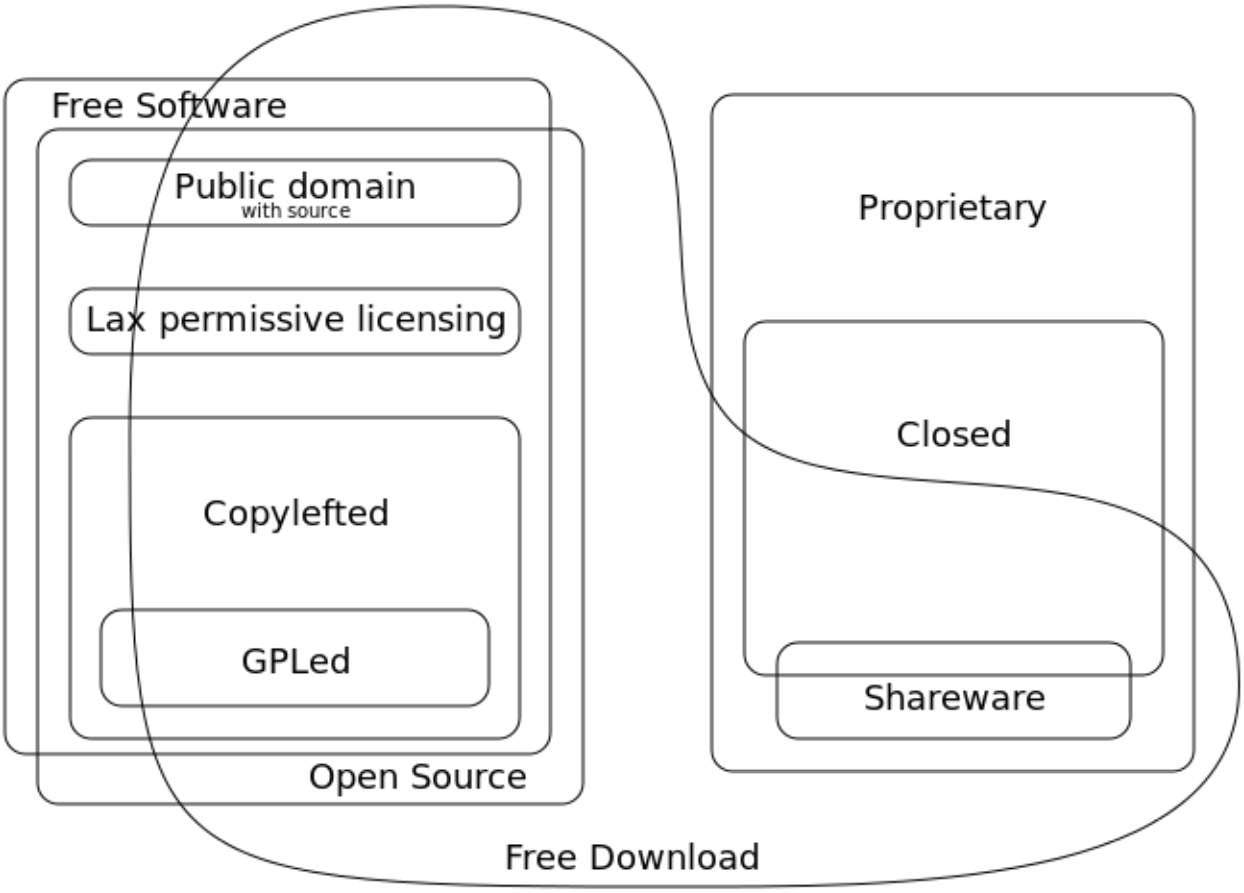
9) La licenza non deve contaminare gli altri programmi. La licenza non deve porre limitazioni su altro software che venga distribuito insieme con il software in licenza. Per esempio, la licenza non deve asserire che tutti gli altri programmi distribuiti sullo stesso supporto devono essere software open source.

10) Conformità. Noi pensiamo che la Open Source Definition racchiuda cio' che la maggior parte della comunita' software intendeva originariamente, e

ancora intende, sul termine "Open Source". In ogni caso il termine e' cominciato ad essere ampiamente utilizzato e il suo significato ha perso un po' di precisione. Il marchio Certificato OSI e' il modo che OSI ha di certificare che la licenza sotto la quale il software e' distribuito risponde ai requisito della Open Source definition; il termine generico "Open Source" non puo' conferire questa garanzia, ma noi incoraggiamo ancora una volta l'utilizzo del termine "Open Source" per intendere la conformità alla OSD.

Software proprietario

Il software proprietario ...



Sistema operativo

Il **sistema operativo,** detto anche SO o OS (operating system) è quel software di sistema che rende il computer utilizzabile dagli utenti. I primi computer non avevano un sistema operativo: i programmi venivano eseguiti, uno alla volta, direttamente sull'hardware della macchina e avevano accesso a tutte le sue risorse. Anche molte console per videogiochi del passato (Megadrive, Super Ninendo, Nintendo 64, ..) non avevano un sistema operativo: senza il gioco inserito mostravano una schermata nera e non facevano nulla. Ancora oggi vi sono piattaforme hardware dalla potenza limitata, come per esempio Arduino, che fanno a meno del sistema operativo.

I sistemi operativi moderni permettono di mandare in esecuzione anche più di un programma "contemporaneamente" grazie al multitasking e di rendere fruibili le memorie di massa gestendo i file system. Inoltre, grazie alla memoria virtuale, possono "vedere" più memoria centrale di quella realmente disponibile.

Il **kernel** è quel software che costituisce il nucleo di un sistema operativo, il "cuore", la parte più "profonda" e vicina all'hardware. I kernel monolitici incorporano al loro interno molte funzioni del sistema operativo, come driver e gestori dei file system, mentre altri, con architettura a microkernel, contengono solo il minimo indispensabile, come per esempio il gestore dei processi e demandano le altre funzionalità a dei programmi utente separati. Al di là delle differenze architetturali, il compito fondamentale di un kernel è gestire le risorse hardware e software del computer e garantire ai processi in esecuzione un accesso sicuro e controllato all'hardware. Dato che normalmente può essere eseguito "contemporaneamente" più di un programma, il kernel ha la responsabilità di assegnare una porzione di tempo-macchina e di accesso all'hardware a ciascun programma (multitasking).

Alcuni identificano il sistema operativo col kernel, mentre per altri il sistema operativo comprende, oltre al kernel e ai vari software di sistema, anche tutti quei programmi, di base e applicativi, forniti nel file di installazione.

Panoramica dei sistemi operativi più diffusi

Unix

Verso la fine degli anni '60 i laboratori di ricerca e sviluppo Bell, della società americana di telecomunicazioni AT&T. decisero di realizzare un nuovo sistema operativo per i propri computer. A causa della difficoltà di realizzazione, il progetto tuttavia venne abbandonato. Alcuni ricercatori non erano d'accordo con le decisioni

dell’azienda e decisero di continuare lo sviluppo. Tra questi ricordiamo in particolare **Dennis Ritchie** e **Ken Thompson**. Il nuovo sistema operativo venne infine alla luce e prese il nome di Unix.

La prima versione di Unix era programmata in assembly per minicomputer PDP-7. Gli assembly sono linguaggi a basso livello, ossia molto vicini all’hardware della macchina e pertanto i programmi scritti in tali linguaggi sono difficilmente trasferibili su computer con architetture diverse.

Per ovviare al problema e rendere Unix facilmente portabile su altre architetture, nei primi anni ‘70 Dennis Ritchie e Ken Thompson idearono un nuovo linguaggio di programmazione ad alto livello, ossia indipendente dall’architettura, chiamato C e lo usarono per riscrivere Unix. Nato originariamente per Unix, negli anni il linguaggio C ha avuto un enorme successo, è approdato ovunque e ha avuto molte estensioni e ‘varianti’ (C++, C#, Objective-C, ...). Il C e i linguaggi da esso derivati sono stati usati per scrivere ogni genere di programma: sistemi operativi, interpreti dei comandi, editor, software di produttività personale, driver per i dispositivi hardware più disparati, videogiochi e perfino interpreti per altri linguaggi di programmazione.

Unix possedeva tutte le caratteristiche richieste da un sistema operativo moderno. Inoltre, grazie al fatto che era programmato ad alto livello, era portabile su architetture diverse. Poiché la AT&T era un’azienda di telecomunicazioni e per una legge antitrust degli USA non poteva commercializzare software, la licenza di Unix era molto permissiva. Era possibile richiedere e ottenere i sorgenti in C del sistema operativo, modificarli e creare delle varianti come fece L’Università di Berkeley che realizzò il suo sistema operativo Unix, conosciuto col nome di Berkeley Software Distribution (BSD). Perfino Microsoft, azienda famosa per aver realizzato Windows, aveva la sua versione di Unix chiamata Xenix. La Silicon Graphics, società un tempo leader del mercato delle workstation grafiche e famosa per aver realizzato la computer grafica di film come Terminator 2 e Jurassic Park, realizzò la sua versione di Unix denominata IRIX.

Jurassic Park - It’s a Unix system!

Grazie alla sua licenza, Unix veniva usato e studiato nelle università di tutto il mondo. Col passare del tempo l’hardware dei computer costava sempre meno e si diffondeva ovunque mentre il software diventava sempre più complesso. A cavallo degli anni ‘80 molte aziende facevano affari d’oro vendendo i propri programmi coperti da copyright. Una licenza con copyright impone molte restrizioni sull’uso che si può fare del programma, per esempio proibisce di modificarlo o di installarlo su più di un computer alla volta. La AT&T decise di rendere Unix un prodotto commerciale aumentandone il prezzo e smettendo di distribuire i sorgenti. A seguito di questa scelta il Dipartimento di Giustizia degli USA impose lo smembramento della società in diverse sotto-compagnie. Le università rimasero ‘orfane’ di Unix perché, pur continuando ad usarlo, non potevano più studiarlo né modificarlo.

GNU

Richard Stallman [https://it.wikipedia.org/wiki/Richard_Stallman], un fisico laureato ad Harvard, lavorava come programmatore al Massachusetts Institute of Technology (MIT), oltre a risiedere stabilmente nel suo ufficio del campus come uno squatter. Nel 1980 venne installata nei laboratori di intelligenza artificiale, dove Stallman lavorava, una nuova stampante di rete laser della Xerox che serviva

numerosi piani dell'edificio. Il software (driver) della stampante era carente di molte funzioni, per esempio non avvisava gli utenti del completamento della stampa né li avvertiva dell'inceppamento della carta, e così si creavano lunghe code di persone in attesa. Stallman chiese alla Xerox il sorgente dei driver per poter aggiungere le funzioni che mancavano ma la richiesta fu rifiutata dalla società che preferiva mantenere nascosti i sorgenti dei propri programmi. Stallman non si diede per vinto: riscrisse il driver con le funzionalità che mancavano e lo distribuì liberamente insieme ai sorgenti in modo che tutti potessero usarlo e migliorarlo. Sempre più convinto della necessità che il software fosse libero come lo era un tempo, Stallman fondò la **Free Software Foundation** [Free Software Foundation] per fornire una infrastruttura legale ai programmatori di **software libero**. Introdusse anche il concetto di **copyleft** e, insieme ad alcuni avvocati, ideò la licenza GNU General Public License (GPL). Il progetto più ambizioso, tuttavia era la creazione di un sistema operativo compatibile con Unix e totalmente libero da copyright chiamato GNU.

*Il **progetto GNU** è una iniziativa collaborativa creata il 27 settembre 1983 da Richard Stallman per creare GNU: un sistema operativo Unix-like completo, utilizzando solo software libero (licenza GPL). Il nome GNU è l'acronimo ricorsivo di "GNU's Not Unix".* [https://it.wikipedia.org/wiki/Progetto_GNU]

Grazie al progetto GNU, i più importanti programmi per Unix, come per esempio il compilatore C GCC (GNU C Compiler) o l'interprete dei comandi bash (Bourne Again Shell) vennero riscritti sotto licenza GPL (libera e copyleft). Tutti questi programmi funzionavano perfettamente su Unix, ma per avere un sistema operativo completo totalmente libero mancava un kernel. Non essendo disponibile un kernel libero, gli sviluppatori del progetto GNU nel 1990 iniziarono a implementarne uno chiamato HURD: a causa della complessità del progetto e della mancanza di sviluppatori, ancora oggi HURD non è considerato pronto per l'uso quotidiano.

GNU/Linux

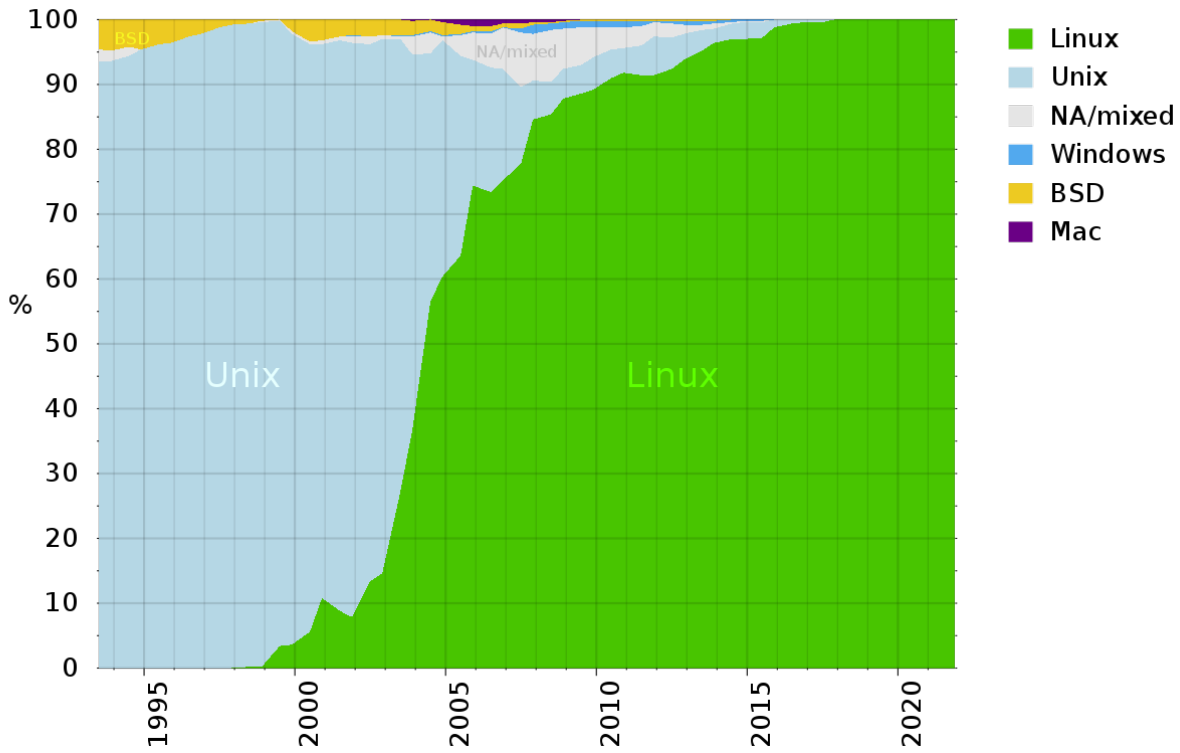
Linux è un kernel monolitico compatibile con Unix originariamente sviluppato nella sua cameretta da Linus Torvalds, uno studente di Informatica finlandese. La prima versione è del 1991. Linux è attualmente in pieno sviluppo grazie al contributo di molti programmatori, mentre il suo ideatore ne dirige tuttora il progetto. Il nome Linux deriva da Linus con la X di Unix. Integrando Linux col software del progetto GNU si ottenne per la prima volta un sistema operativo compatibile con Unix chiamato **GNU/Linux** o, più semplicemente ma meno correttamente, Linux. Da

notare che in origine il kernel del progetto GNU doveva essere HURD. Ovviamente anche Linux è software libero ed è anche grazie a ciò si deve la sua ampia diffusione.

Col passare del tempo nacquero numerosi sistemi operativi basati sui software del progetto GNU e Linux: le cosiddette **distribuzioni GNU/Linux** o **distribuzioni Linux**. Le distribuzioni GNU/Linux sono moltissime perché da una ne possono derivare tantissime. Per esempio, da Debian deriva Ubuntu, da Ubuntu deriva Mint. L'immagine reperibile a questo indirizzo https://upload.wikimedia.org/wikipedia/commons/e/e5/GNU-Linux_distro_timeline_10_9.png, troppo grande per essere inserita negli appunti, fornisce un'idea piuttosto dettagliata di come si siano sviluppate nel tempo e quale sia il legame di parentela tra le varie distribuzioni GNU/Linux. Come si evince dalla figura, quasi tutte le distribuzioni derivano da Debian (che avete visto numerose volte a lezione), Slackware o Red Hat. Poiché Linux è un kernel monolitico che accentra in sé numerose funzioni, qualcuno considera Linux un sistema operativo. La correttezza di questa affermazione dipende dalla definizione di sistema operativo che adottiamo. Si tenga comunque presente che neanche un kernel monolitico come Linux è sufficiente a rendere un computer utilizzabile da parte dell'utente finale senza del software aggiuntivo, come quello del progetto GNU.

N.B. Sebbene sia consuetudine definire Debian una distribuzione GNU/Linux, ciò non è del tutto esatto in quanto può utilizzare anche altri kernel, come GNU Hurd o il kernel di FreeBSD.

Attualmente le distribuzioni GNU/Linux vengono utilizzate nella maggior parte dei server e dei supercomputer.



Utilizzo di Linux nei supercomputer [https://it.wikipedia.org/wiki/Supercomputer]

Android

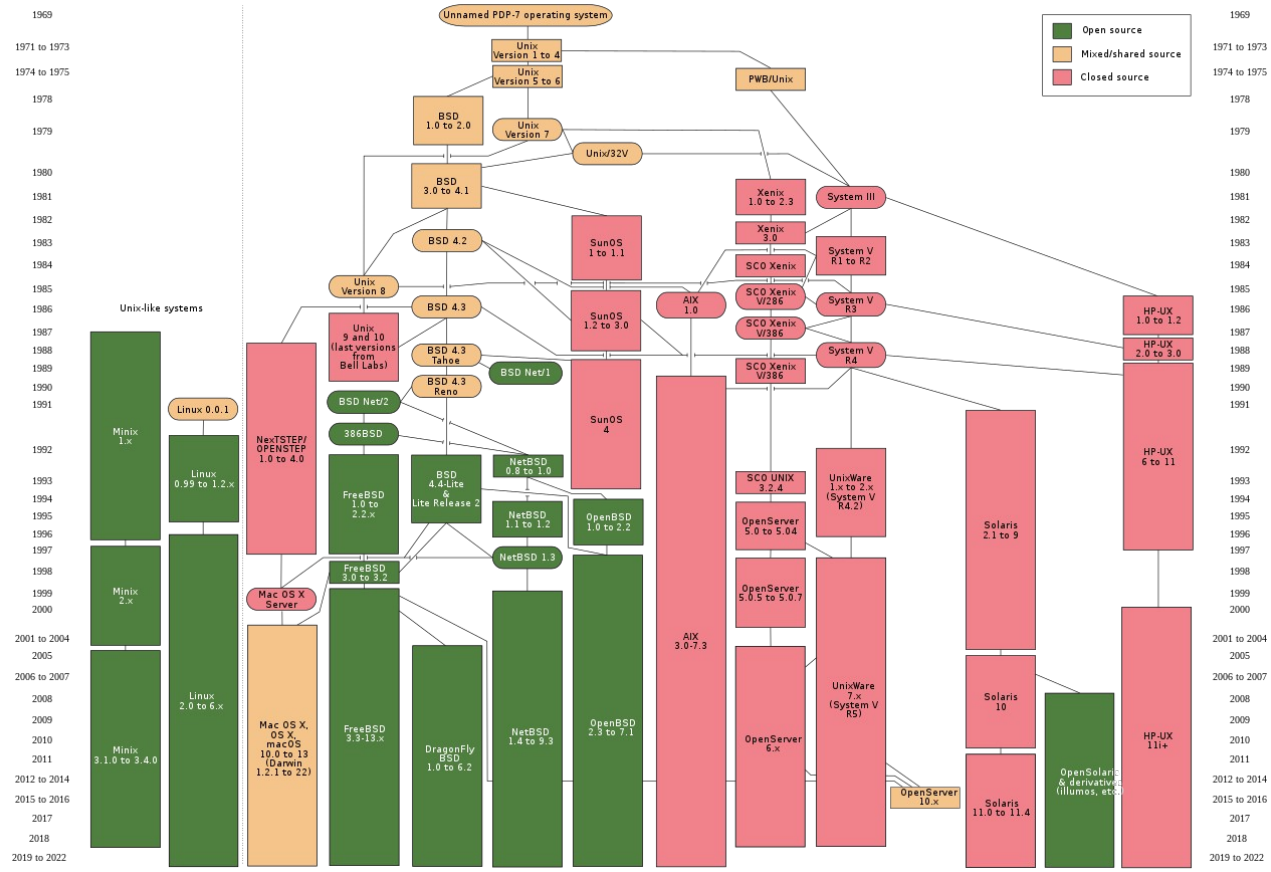
Anche **Android**, diffusissimo in smartphone e tablet, usa Linux come kernel, tuttavia *si differenzia radicalmente dal sistema operativo GNU/Linux poiché il componente GNU è in effetti minimo.* [https://www.gnu.org/philosophy/android-and-users-freedom.it.html]

Chrome OS

Un altro sistema operativo oggi piuttosto diffuso è **Chrome OS** di Google, derivato da Gentoo, una distribuzione GNU/Linux e preinstallato soprattutto sui portatili Chromebook.

Mac OS X, macOS e iOS

Dalla decima versione in poi, anche il sistema operativo di Apple è basato sul kernel di FreeBSD, uno Unix. La X presente nel nome Mac OS X, oltre a indicare il numero di versione è un chiaro riferimento a ciò. Nelle ultime versioni del sistema operativo il nome è cambiato in macOS. Anche iOS, installato sui cellulari della Apple è a sua volta uno Unix.



Di Eraserhead1, Infinity0, Sav_vas - Levenez Unix History Diagram, Information on the history of IBM's AIX on ibm.com, CC BY-SA 3.0, [https://commons.wikimedia.org/w/index.php?curid=1801948]

Windows

Prima che pensiate che tutti i sistemi operativi al mondo derivino da Unix, è doveroso accennare all'esistenza di una alternativa piuttosto diffusa: Windows.

Windows è un termine generico assegnato ai diversi sistemi operativi programmati da Microsoft (abbreviato MS). La versione più recente del sistema operativo è la numero 11 ma non fatevi trarre in inganno dal numero perché il modo in cui MS assegna le versioni ai propri sistemi operativi non è normale (vale anche per i nomi delle console).

In origine (anni '90) vi erano due famiglie distinte di sistemi operativi Windows: la famiglia NT, compreso Windows 2000, per gli utenti 'business' e Windows 9x (95, 98, millenium) per i poveracci (su cui però giravano i giochi). In seguito MS decise di unificare le due famiglie, o meglio di proseguire solo con la famiglia NT inserendo caratteristiche della famiglia 9x in un unico sistema operativo chiamato Windows XP. Dopo Windows XP, MS farà uscire Windows Vista, Windows 7, Windows 8, Windows 10, Windows 11. Se ve lo state chiedendo, la risposta è **no**, non è una

dimenticanza: **Windows 9 non esiste** e il motivo lo sanno solo gli addetti al marketing di MS.

Esisteva una versione di Windows anche per smartphone e tablet ma purtroppo MS ne ha interrotto lo sviluppo. Versioni modificate di Windows girano anche sulle Xbox, le console prodotte da MS.

N.B. Il primo sistema operativo di Microsoft (comprato) non è una versione di Windows bensì un DOS (Disk Operating System) e le prime versioni di Windows altro non erano che programmi per MS-DOS.

Sistemi operativi alternativi

I sistemi operativi di Microsoft sono i più usati sui personal computer mentre tra i server sono più diffusi sistemi UNIX/Unix-like, in genere distribuzioni GNU/Linux. Sono tuttavia esistiti ed esistono tuttora moltissimi sistemi operativi totalmente estranei sia a Windows che a Unix, come Haiku (clone di BeOS), Aros (clone di AmigaOS), MenuetOS, TempleOS (il sistema operativo del tempio), ... Spesso questi sistemi operativi sono sviluppati da pochi o addirittura da un solo programmatore per pura sfida intellettuale, o per il desiderio di far rivivere un sistema operativo del passato. In questa pagina https://it.wikipedia.org/wiki/Lista_di_sistemi_operativi trovate un elenco di tutti(?) i sistemi operativi presenti e passati.

File system

*Il **file system** (in acronimo FS), in Informatica, indica informalmente un meccanismo con il quale i file sono posizionati e organizzati su dispositivi informatici utilizzati per l'archiviazione dei dati, ad esempio unità di memoria di massa (come unità a nastro magnetico, dischi rigidi, dischi ottici, unità di memoria a stato solido o, in casi particolari, anche nella memoria centrale) o su dispositivi remoti tramite protocolli di rete.* [https://it.wikipedia.org/wiki/File_system]

Quel componente software del sistema operativo che si occupa di organizzare e rendere fruibili le memorie mediante operazioni come elencazione, cancellazione, copia, spostamento, … prende il nome di **gestore del file system**.

Vi sono numerosi tipi di file system, ognuno coi suoi pregi e difetti. I più usati sono:

- FAT32, un FS usato da MS originariamente per il DOS e per le versioni 9x di Windows e attualmente utilizzato nelle memorie esterne, come le chiavette USB, per via della sua compatibilità con qualunque sistema operativo. Ha

grossi limiti sulla dimensione dei file ed è case insensitive, ossia non distingue tra maiuscole e minuscole.

- NTFS, un FS proprietario usato da MS per la famiglia NT usato dalle versioni moderne di Windows. Il suo formato è chiuso (MS non rende note le specifiche) e a causa di ciò fino a qualche anno fa Linux era in grado di leggerlo ma non di scriverlo. Attualmente è pienamente supportato anche da Linux.
- EXT4, ultimo di una serie di FS usati di default nella maggioranza delle distribuzioni GNU/Linux. Pur avendo un formato aperto, fino a non molto tempo fa non era leggibile né scrivibile da Windows se non tramite programmi di terze parti.

File

Un **file** è un archivio di dati a cui viene associato un percorso (path).

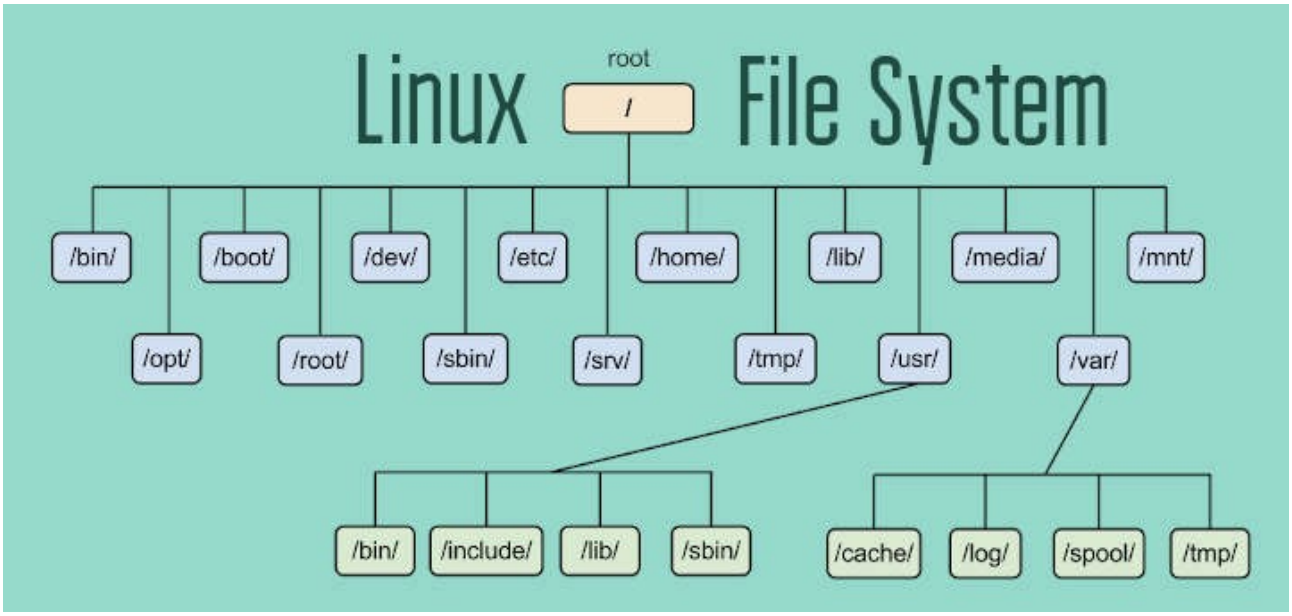
Directory

Una directory (cartella) è un file contenente nomi di altri file ossia un contenitore di file.

Partizione

*Una **partizione**, in informatica indica una suddivisione logica di un'unità di memorizzazione fisica (tipicamente una memoria di massa come un disco rigido o una chiavetta USB). Le singole unità logiche vengono viste dal sistema operativo come unità separate e possono essere formattate e gestite in maniera del tutto indipendente.* [https://it.wikipedia.org/wiki/Partizione_%28informatica%29] Il partizionamento può servire a molti scopi, per esempio ad installare due sistemi operativi nella stessa memoria fisica (utilissimo nel caso dei portatili) o a tenere i dati separati dal sistema operativo (così se lo si reinstalla non bisogna ricopiarli). Ogni unità di memoria deve possedere almeno una partizione. Una partizione viene formattata con un solo file system.

Generalmente, anche se non sempre, i file system possiedono una struttura gerarchica anche detta “ad albero” (la rappresentazione generalmente viene fatta al rovescio, con la radice in alto). Vi è una cartella radice (root) che può contenere file e cartelle che, a loro volta possono contenere file e cartelle e così via, virtualmente all'infinito (in realtà esistono dei limiti che dipendono dal tipo di file system). In figura viene riportata la struttura del file system (solo directory) di una distribuzione GNU/Linux.



Linux FS [https://www.tecmint.com/wp-content/uploads/2012/07/Linux-File-System.jpg]

Al posto di un albero si può anche pensare ad una scatola che può contenere oggetti e scatole, che a loro volta possono contenere oggetti e scatole e così via.

La struttura ad albero è piuttosto intuitiva e flessibile e per questo viene adottata dalla quasi totalità dei file system moderni, anche se esistono delle eccezioni, come i file system usati dalle console per archiviare i salvataggi nelle memory card (tutti i salvataggi insieme, senza cartelle) o quello di BSCW (a grafo diretto aciclico).

Nei sistemi operativi derivati da Unix la radice (root in inglese) si indica col simbolo / (detto barra o slash). Lo stesso simbolo viene utilizzato per separare cartelle e file nel formare il nome di un percorso, per esempio /home/utente/Immagini/vacanze.jpg . In molte distribuzioni GNU/Linux le partizioni delle unità di memoria rimovibili vengono montate nella directory /media mentre quelle permanenti in genere nella directory /mnt .

A differenza dei sistemi derivati da Unix, in Windows non vi è una sola radice ma molte radici (ognuna col proprio albero) quante sono le partizioni montate. Le radici vengono indicate con una lettera maiuscola dell'alfabeto seguita da : (per esempio C: , D: , E: , ...). Se per esempio si inserisce una chiavetta USB (opportunamente formattata), nelle ‘risorse del computer’ comparirà una nuova ‘unità’ (anche se le partizioni leggibili sono più di una ... no comment!): cliccando sulla lettera corrispondente si accederà alla radice di quel file system.

In Windows un esempio di percorso potrebbe essere C:\Documents and Settings\ utente\Documents . Da notare che come separatore viene usato il simbolo \ (detto barra rovesciata o backslash).

Operazioni sul file system

Con l'operazione di **copia**, un file (o directory) viene duplicato in un'altra porzione di memoria. Con l'operazione di **spostamento**, il file cambia il proprio percorso ma occupa sempre la medesima porzione di memoria. Con l'operazione di **cancellazione**, il riferimento al file viene cancellato e la memoria occupata dal file viene resa disponibile per essere sovrascritta. Dal menù contestuale è possibile copiare, spostare e cancellare file:

- Se si sceglie la voce cancella il file viene cancellato;
- Se si copia e si incolla il file, esso viene copiato;
- Se si taglia e si incolla il file tra cartelle della stessa partizione, il file viene spostato, ossia gli viene cambiato percorso;
- Se si taglia e si incolla il file tra cartelle di partizioni diverse (della stessa unità o di unità differenti), il file viene copiato e poi cancellato;
- Se si trascina il file tra due cartelle della stessa partizione, il file viene spostato;
- Se si trascina il file tra carelle di partizioni diverse (della stessa unità o di unità differenti), il file viene copiato.

Multitasking

Un programma che risiede nella memoria centrale (RAM) del computer, in esecuzione o in procinto di essere eseguito, viene detto **processo**. Il **gestore dei processi** (**scheduler**) del sistema operativo carica il programma, in tutto o in parte, da una memoria di massa (hard disk, solid state drive, chiave USB, …) o dalla rete, alla memoria centrale.

Con **multitasking** si intende l’esecuzione più o meno “contemporanea” di più programmi.

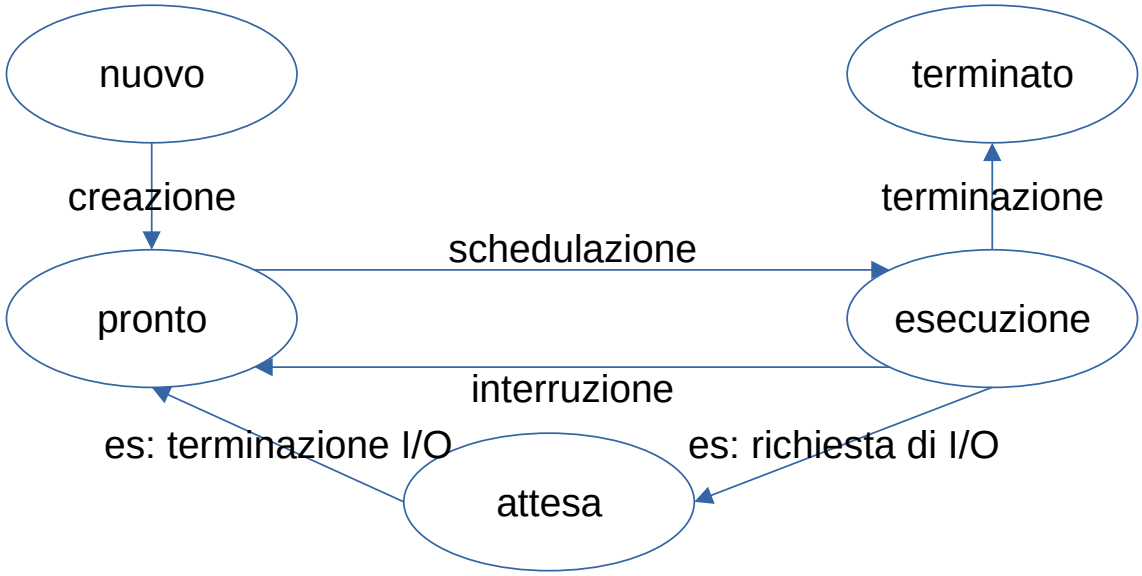
N.B. In un dato istante ci può essere realmente in esecuzione un solo programma per ogni processore/core.

Nel **multitasking senza prelazione** (non preemptive), il passaggio da un processo all’altro avviene **solo** quando il processo termina la sua esecuzione o è in attesa del completamento di una operazione di input/output (per esempio la lettura o la scrittura da o su una memoria di massa).

Nel **multitasking con prelazione** il passaggio da un processo all’altro avviene anche allo scadere del tempo assegnato al processo. Il tempo assegnato al processo può

variare in base alla sua priorità ma comunque il sistema operativo alterna i processi con tale rapidità che la loro esecuzione appare simultanea.

Hardware e sistemi operativi moderni supportano il multitasking con prelazione.



Avanzamento dei processi in un sistema dotato di multitasking con prelazione

Memoria virtuale

La **memoria virtuale** è un meccanismo del sistema operativo che permette l'esecuzione di programmi e l'elaborazione di dati, più grandi della memoria centrale disponibile. Dà l'illusione ai processi di avere spazio potenzialmente illimitato e di poter elaborare file di dimensioni enormi. Questo risultato all'apparenza miracoloso si ottiene grazie al **gestore della memoria virtuale** che utilizza la memoria di massa (in genere una memoria interna, possibilmente un solid state drive perché è veloce) e sfruttando il principio di località spaziale.

Il principio di località spaziale afferma che in un certo istante l'esecuzione di un programma è localizzata, ossia limitata a una parte circoscritta del codice. Per esempio, finché siete nel menù di configurazione di un gioco, il codice del gameplay vero e proprio non viene eseguito.

Il sistema operativo non carica il programma da eseguire e i suoi dati tutto in una volta ma si limita a quei "pezzi" (pagine) che servono in un dato istante. Quando lo spazio nella memoria centrale si esaurisce, viene liberata la memoria occupata da quei "pezzi" che si presume non servano a breve. La scelta di quale porzioni di

memoria liberare dipende da vari fattori: si possono scegliere per esempio quelle porzioni che non vengono usate da molto tempo.

La memoria virtuale ha una controindicazione: anche utilizzando come supporto un drive allo stato solido, che è la memoria di massa più veloce attualmente disponibile, la sua velocità è comunque migliaia di volte inferiore a quella della memoria centrale: l'accesso frequente alla memoria di massa causerà degli inevitabili rallentamenti nell'esecuzione del programma.