Informatica per Statistica
Riassunto della lezione del 03/10/2012

Igor Melatti

Il sistema binario e la rappresentazione di informazioni

- Tra le varie attività che gli competono, un computer ha anche necessità di memorizzare *informazioni*
- È chiaro che, per farlo, è necessario *rappresentare* tali informazioni in qualche modo
- All'interno di un computer, per motivi ingegneristici, ogni informazione viene rappresentata come sequenza (talvolta chiamata *stringa*) di 0 ed 1 (chiamati *bit*)
  - numeri di vari tipi
    - interi senza segno (notazione binaria)
    - interi con segno (notazione binaria in complemento a due)
    - numeri reali a precisione finita (standard IEEE 754, che combina notazione binaria con la notazione mantissa–esponente)
  - caratteri alfanumerici, comprensivi di punteggiatura (ASCII, Unicode)
  - immagini (JPG, PNG, GIF, ...)
  - video (AVI, MPEG, ...)
  - testi formattati, ovvero con interruzioni di pagina, tabelle, grassetti, font, ... (DOC, ODT, ...)
  - banche dati (MDB, ODB, ...)
  - sequenze di istruzioni in linguaggio macchina (EXE, ELF, ...)
  - ...
- Nel rappresentare stringhe di bit, spesso si usano la base ottale e l'esadecimale, come vedremo
- Per quanto riguarda i numeri naturali (interi positivi)

- si usa quella che viene chiamata *notazione posizionale*
- da notare che la *notazione decimale*, usata ogni giorno per rappresentare i numeri in tutto il mondo, non è altro che un caso particolare della notazione posizionale
- le notazioni binaria, ottale ed esadecimale che verranno esposte qui di seguito sono altri casi particolari della notazione posizionale
- l'idea della notazione posizionale è quella di partire da una *base* $b > 1$ (10 per la notazione decimale, 2 per quella binaria, 8 per quella ottale, 16 per quella esadecimale) e da dei simboli elementari chiamati *cifre*
  - le cifre dovranno essere esattamente b
  - usando le cifre arabo-indiane, si va da 0 a $b - 1$
  - se $b > 10$, allora si usano altri simboli, ad es. lettere maiuscole
  - infatti, nella notazione esadecimale si usano, come cifre, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, D, C, E, F
- con questa premessa, è possibile rappresentare qualsiasi numero come sequenza di cifre
- se si vuole conoscere il valore di una sequenza di cifre $c_n \dots c_0$ (ovvero, data una rappresentazione, trovare il numero rappresentato), esso può essere trovato applicando la seguente formula

$$\text{Valore}(c_n \dots c_0) = \sum_{i=0}^{n} b^i c_i$$

- per esempio, il valore di 123_{10} è $\text{Valore}(123_{10}) = \sum_{i=0}^{2} 10^i c_i = 3 \cdot 10^0 + 2 \cdot 10^1 + 1 \cdot 10^2 = 3 + 20 + 100 = 123$
- con il sistema binario, si hanno solo le cifre binarie 0 e 1, dette *bit* dall'inglese *b(inary dig)it*
- vale sempre la stessa formula di sopra per conoscere il valore di una stringa di bit
- ad esempio, il valore di 1111011_2 è $\text{Valore}(1111011_2) = \sum_{i=0}^{6} 2^i c_i = 1{\cdot}2^0{+}1{\cdot}2^1{+}0{\cdot}2^2{+}1{\cdot}2^3{+}1{\cdot}2^4{+}1{\cdot}2^5{+}1{\cdot}2^6 = 1{+}2{+}8{+}16{+}32{+}64 = 123$
- tale regola può quindi essere usata, in pratica, per convertire un numero (intero e positivio) da binario a decimale
- per fare il passo inverso, si può usare l'algoritmo 1 per convertire un numero dalla rappresentazione in base 2 a quella in base 10
- l'algoritmo 1 non dà nessun output per il caso in cui $n = 0$
- può essere corretto, ottenendo l'algoritmo 2
- ecco un esempio di come possa essere applicato l'algoritmo 2 per trovare la rappresentazione in base 2 di 123_{10}

```
1 DaDecimaleABinario(n ∈ ℕ)
2   H = n.
3   Finché  H ≠ 0 ripeti i seguenti passi:
4     K = H mod 2, H = H div 2.
5     Dai in output il bit K,
6       posizionandolo a sinistra del bit precedente.
```

Figure 1: Conversione da Decimale a Binario

```
1 DaDecimaleABinarioMigliore(n ∈ ℕ)
2 H = n.
3 Ripeti i seguenti passi:
4   K = H mod 2, H = H div 2.
5   Dai in output il bit K,
6     posizionandolo a sinistra del bit precedente.
7 Finché  H ≠ 0
```

Figure 2: Conversione da Decimale a Binario (versione più completa)

	$H = 123$	output $= \emptyset$
$K = 123 \bmod 2 = 1$	$H = 123 \operatorname{div} 2 = 61$	output $= 1_2$
$K = 61 \bmod 2 = 1$	$H = 61 \operatorname{div} 2 = 30$	output $= 11_2$
$K = 30 \bmod 2 = 0$	$H = 30 \operatorname{div} 2 = 15$	output $= 011_2$
$K = 15 \bmod 2 = 1$	$H = 15 \operatorname{div} 2 = 7$	output $= 1011_2$
$K = 7 \bmod 2 = 1$	$H = 7 \operatorname{div} 2 = 3$	output $= 11011_2$
$K = 3 \bmod 2 = 1$	$H = 3 \operatorname{div} 2 = 1$	output $= 111011_2$
$K = 1 \bmod 2 = 1$	$H = 1 \operatorname{div} 2 = 0$	output $= 1111011_2$

- per rappresentare un numero n in base 2, occorrono $\lfloor \log_2 n \rfloor + 1$ bits
  - per un numero (reale) r, fare $\lfloor r \rfloor$ vuol dire prendere il più grande intero minore od uguale ad r (in pratica, si arrotonda sempre all'intero più basso, e se è già intero non occorre far nulla), quindi $\lfloor 3.999999 \rfloor = 3$, $\lfloor 10.000001 \rfloor = 10$, $\lfloor 4 \rfloor = 4$, ...
  - analogamente, per un numero (reale) r, fare $\lceil r \rceil$ vuol dire prendere il più piccolo intero maggiore od uguale ad r (in pratica, si arrotonda sempre all'intero più alto, e se è già intero non occorre far nulla), quindi $\lceil 3.999999 \rceil = 4$, $\lceil 10.000001 \rceil = 11$, $\lceil 4 \rceil = 4$, ...
  - quindi $\lfloor r \rfloor + 1 = \lceil r \rceil$ se r non è intero, altrimenti $\lfloor r \rfloor = \lceil r \rceil$ se r è intero
- e servono $\lfloor \log_{10} n \rfloor + 1$ cifre decimali per rappresentarlo in base 10
  - in generale, servono $\lfloor \log_k n \rfloor + 1$ cifre in base k per rappresentare n in base k

- dato che $\log_2 n > \log_{10} n$, ci possono volere troppi bit per rappresentare numeri relativamente piccoli
    - per rappresentare un milione, occorrono 20 bit (e solo 7 cifre decimali)
- è possibile usare le basi 8 e 16 (non a caso potenze di 2...) per accorciare le rappresentazioni di numeri in binario, usando una conversione diretta
    - per rappresentare un milione, occorrono 20 bit
    - con la notazione in base 16, si riduce la lunghezza delle rappresentazioni di un fattore 4 (perché $2^4 = 16$...); con la notazione in base 8, si riduce la lunghezza delle rappresentazioni di un fattore 3 (perché $2^3 = 8$...)
    - quindi, occorrono $\lceil 20/4 \rceil = 5$ cifre esadecimali per rappresentare un milione, mentre occorrono $\lceil 20/3 \rceil = 7$ cifre ottali per lo stesso numero
    - per la conversione da base 2 a base 16, basta dividere il numero in pezzetti da 4 bit, per poi convertire singolarmente ciascuna cifra
        - quindi, 0000_2 diventa 0_{16}, 0001_2 diventa 1_{16}, ..., 1001_2 diventa 9_{16}, 1010_2 diventa A_{16}, 1011_2 diventa B_{16}, 1100_2 diventa C_{16}, 1101_2 diventa D_{16}, 1110_2 diventa E_{16}, 1111_2 diventa F_{16}
    - esempio: $10011010101111100100001_2 = 26AF91_{16} = 11527621_8$, dal momento che, suddividendo a gruppi di 4, si ha 0010 0110 1010 1111 1001 0001, e $0001_2 = 1_{16}$, $1001_2 = 9_{16}$, $1111_2 = F_{16}$, $1010_2 = A_{16}$, $0110_2 = 6_{16}$ e $0010_2 = 2_{16}$
    - per la conversione inversa, basta trasformare ogni cifra esadecimale in un gruppo di 4 bit
        - quindi, 0_{16} diventa 0000_2, 1_{16} diventa 0001_2, ..., 9_{16} diventa 1001_2, A_{16} diventa 1010_2, B_{16} diventa 1011_2, C_{16} diventa 1100_2, D_{16} diventa 1101_2, E_{16} diventa 1110_2, F_{16} diventa 1111_2
    - esempio: $FA3_{16} = 111110100011_2$, dal momento che $3_{16} = 0011_2$, $A_{16} = 1010_2$ e $F_{16} = 1111_2$.
    - per la base 8, vale lo stesso discorso, ma si usano gruppi di 3 bit

- Caratteri: codice ASCII
    - nel codice ASCII ogni carattere è un *byte*, ovvero una sequenza di 8 bit
    - questo ha fatto sì che il byte diventasse l'unità di misura delle memorie

  - cioè, dato che tutto dentro un computer è sotto forma di bit, anche le memorie (RAM, dischi o altro) sono fatte di bit
  - ovvero, ciò che vi viene "ricordato" sono bit
  - perché, come stiamo per vedere, praticamente ogni cosa è rappresentabile coi bit
  - non si dice però che una memoria può memorizzare 16 bit
  - ma si dice che può memorizzare 2 bytes
  - in realtà, le memorie memorizzano molti bytes
  - limitandosi alla RAM e al disco: le dimensioni tipiche sono di miliardi di bytes, e di centinaia di miliardi di bytes, rispettivamente
  - siccome parlare di migliaia di miliardi e cose simili è un po' scomodo, si usano le sigle
  - già nel Sistema Metrico Internazionale ci sono dei prefissi (vedere Figura 3, tratta da Wikipedia), ma sono poco usati (mai sentito parlare di gigametri?)
  - nell'informatica, se ne fa un grand'uso, ma il loro valore è leggermente diverso (vedere Figura 4, ancora tratta da Wikipedia)
  - 1 KB vale $1024 = 2^{10}$ bytes, 1 MB vale $1048576 = 2^{20}$ bytes, e così via
  - il codice ASCII può essere riassunto come in Figura 5
    * ciascun carattere è rappresentato su un byte, ovvero 8 bit
    * quindi servono 2 cifre esadecimali per rappresentare un carattere
    * nella tabella del codice ASCII data in Figura 5, l'intestazione della riga dà la prima cifra esadecimale, quella della colonna la seconda
    * le prime 2 righe (quindi 32 caratteri) sono quelli non (direttamente) stampabili
    * tra gli altri, da notare il carattere **LF** (*line feed*, corrispondente a $0A_{16}$), che rappresenta l'andata a capo (tasto Return o Invio)
  - ASCII fatto apposta per semplificare alcune operazioni: tutti i caratteri vicini, facile il passaggio alle maiuscole

- Caratteri: standard UNICODE e codifica UTF-8
  - l'ASCII fa il minimo indispensabile (niente lettere accentate, niente lettere greche, ...)
  - nello UNICODE, tutti i caratteri delle lingue conosciute al mondo trovano un posto (anche arabo, cinese, giapponese, cirillico, Braille...)
  - per rappresentare lo UNICODE con stringhe di bit si usa per lo più il codice UTF-8 (usato da quasi tutte le pagine del WWW)

Prefissi del Sistema Internazionale

10^n	Prefisso	Simbolo	Nome	Equivalente decimale
10^{24}	yotta	Y	Quadrilione	1 000 000 000 000 000 000 000 000
10^{21}	zetta	Z	Triliardo	1 000 000 000 000 000 000 000
10^{18}	exa	E	Trilione	1 000 000 000 000 000 000
10^{15}	peta	P	Biliardo	1 000 000 000 000 000
10^{12}	tera	T	Bilione	1 000 000 000 000
10^{9}	giga	G	Miliardo	1 000 000 000
10^{6}	mega	M	Milione	1 000 000
10^{3}	kilo	k	Mille	1 000
10^{2}	hecto	h	Cento	100
10^{1}	deca	da	Dieci	10
10^{0}			Uno	1
10^{-1}	deci	d	Decimo	0,1
10^{-2}	centi	c	Centesimo	0,01
10^{-3}	milli	m	Millesimo	0,001
10^{-6}	micro	µ	Milionesimo	0,000 001
10^{-9}	nano	n	Miliardesimo	0,000 000 001
10^{-12}	pico	p	Bilionesimo	0,000 000 000 001
10^{-15}	femto	f	Biliardesimo	0,000 000 000 000 001
10^{-18}	atto	a	Trilionesimo	0,000 000 000 000 000 001
10^{-21}	zepto	z	Triliardesimo	0,000 000 000 000 000 000 001
10^{-24}	yocto	y	Quadrilionesimo	0,000 000 000 000 000 000 000 001

Figure 3: Prefissi del sistema Metrico Internazionale

IEC prefix		Representations				Customary prefix	
Name	Symbol	Base 2	Base 1024	Value	Base 10	Name	Symbol
kibi	Ki	2^{10}	1024^1	1 024	$\approx 1.02 \times 10^3$	kilo	k, K
mebi	Mi	2^{20}	1024^2	1 048 576	$\approx 1.05 \times 10^6$	mega	M
gibi	Gi	2^{30}	1024^3	1 073 741 824	$\approx 1.07 \times 10^9$	giga	G
tebi	Ti	2^{40}	1024^4	1 099 511 627 776	$\approx 1.10 \times 10^{12}$	tera	T
pebi	Pi	2^{50}	1024^5	1 125 899 906 842 624	$\approx 1.13 \times 10^{15}$	peta	P
exbi	Ei	2^{60}	1024^6	1 152 921 504 606 846 976	$\approx 1.15 \times 10^{18}$	exa	E
zebi	Zi	2^{70}	1024^7	1 180 591 620 717 411 303 424	$\approx 1.18 \times 10^{21}$	zetta	Z
yobi	Yi	2^{80}	1024^8	1 208 925 819 614 629 174 706 176	$\approx 1.21 \times 10^{24}$	yotta	Y

Figure 4: Prefissi del sistema Metrico Internazionale nel caso binario

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

Figure 5: Il codice ASCII

  - qui basti sapere che nell'UTF-8 i caratteri "semplici" (quelli dell'ASCII) hanno la stessa codifica che avevano in ASCII
  - per gli altri, si usano 2 o più bytes
  - esistono però altri metodi
  - un file di testo è fatto di soli caratteri, seconda una delle possibili codifiche; se un programma usa un codice mentre un file è stato scritto con l'altro, i caratteri speciali verranno mal visualizzati

- Immagini, suoni, video: le cose si complicano
  - l'acquisizione di tali informazioni avviene tramite una coppia *trasduttore–convertitore*
  - il trasduttore prende dati dalla realtà esterna (suoni come onde sonore, immagini come luce) e li trasforma in una sequenza di tensioni, il convertitore trasforma tali tensioni in numeri, rappresentati come bit
  - tipicamente, tutto ciò lo fa un apparecchio solo (fotocamera digitale, videocamera, computer dotato di microfono ed opportuno software...)
  - nel caso di un'immagine, c'è di solito un solo istante in cui si riprende la realtà
  - nel caso di video o anche solo suoni, c'è una sequenza di istanti
    * per essere perfetti, occorrerebbe avere infiniti punti di acquisizione dell'input
    * ma nessun apparecchio costruito dall'uomo può memorizzare infinite informazioni digitali

    - per fortuna, basta selezionare opportuni istanti, certo molto vicini ma l'informazione risultante può essere effettivamente memorizzata
    - tecnicamente, si dice che occorre *campionare*, opportuni studi dicono quale frequenza occorre usare perché i sensi umani non colgano la differenza
    - esempio: memorizzazione di suoni
        - li si campiona con una frequenza di 44100 Hz (ovvero, in un secondo si prendono 44100 valori per quanto riguarda intensità e altezza)
        - ognuna di queste informazione viene tradotta in numeri, rappresentati in binario con 16 bit
        - questo fa sì che ogni secondo ci siano 88200 numeri di 16 bit, ovvero 176400 bytes
        - questo vuol dire che per memorizzare un'ora di musica occorrono circa 605 MB
        - infatti un normale CD, coi suoi 700 MB, arriva a 74 minuti (quanto dura la nona di Beethoven...)
        - questo formato è detto *wave* (files .wav)
    - esempio: memorizzazione di immagini (singole)
        - ogni macchina fotografica ha una sua risoluzione $r \times c$ (righe per colonne)
        - quindi ha $p = rc$ pixels
        - ovviamente, più sono i pixels (quindi più sono le righe e le colonne), più ciascuno di essi diventa piccolo e l'immagine meno sgranata (a parità di grandezza dell'area dell'immagine)
        - per memorizzare un'immagine nella maniera più naif, si memorizza per ogni pixel la sua colorazione
        - può essere data come RGB (visualizzazione su schermo, ogni pixel ha 3 bytes) o CYMK (stampa, ogni pixel 4 bytes)
        - questo approccio è detto *bitmap* (files .bmp)
    - i video combinano quanto detto prima per immagini e suoni
- rappresentare in questa maniera non è sempre conveniente, soprattutto per immagini e video
- ci sono infatti tecniche di compressione
    - classificate in *loseless* (senza degrado di qualità) e *lossy* (con degrado di qualità)
    - per le immagini sono famosi i casi di GIF e JPEG, rispettivamente
    - per i video, c'è l'MPEG e l'AVI

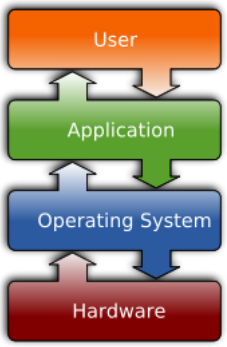

Figure 6: Architettura di un sistema operativo

- Tecniche di compressione possono essere usati su qualsiasi tipo di informazioni
  - le tecniche viste prima sono ottimizzate per immagini e filmati, mentre non funzionano bene su altri tipi di dati
  - programmi come WinZip e gzip, WinRar e analoghi Linux, sono invece fatti per comprimere dati eterogenei
  - utile per trasmettere informazioni (in rete o per la copia su supporti meno capienti degli hard disk)
  - mentre esistono programmi in grado di visualizzare direttamente immagini e filmati compressi, con questi programmi occorre prima "scompattare" il tutto

Il sistema operativo

- Tutto l'hardware del mondo è inutile senza software
  - nella Fig. 6 si vede che tra l'utente (che usa il computer) e l'hardware ci sono 2 strati
  - quello più vicino all'hardware è il *sistema operativo* (Windows, Linux, MacOS-X, ...)
  - quello più vicino all'utente contiene i *programmi applicativi* (semplicemente *applicazioni* o *programmi*) che l'utente usa direttamente
    - programmi per scrivere (Word)
    - fogli di calcolo (Excel)
    - programmi multimediali (Windows Media Player)
    - questi sono per Windows, sia chiaro che esistono i corrispettivi programmi per Linux e MacOS-X

    - altro (videogiochi, programmazione, ...)
  - per far sì che un software possa essere usato, occorre *installarlo*
    - semplificando, vuol dire copiarlo su disco fisso in modo che possa essere propriamente eseguito quando serve
    - tipica sorgente per l'installazione è un file preso da un qualche supporto di input (rete, CD, DVD, chiave USB...)
    - vale sia per l'intero sistema operativo che per i singoli applicativi
    - a seconda della complessità del programma stesso, può essere un'operazione facile o difficile
    - un sistema operativo è tra le cose più difficili da installare
  - usare un software vuol dire *eseguirlo* (o anche avviarlo, o aprirlo, talvolta *lanciarlo*)
    - semplificando, vuol dire prenderlo dal disco fisso e copiarlo (almeno in parte) in RAM
    - dalla RAM può essere *eseguito* (almeno secondo il modello di Von Neumann)
    - se va tutto bene, un software si installa *una sola volta*, e lo si può solitamente eseguire *molteplici volte*
  - il sistema operativo non viene mai modificato direttamente dall'utente "normale"
    - al massimo, si eseguono gli *aggiornamenti* predisposti da chi ha programmato il sistema operativo stesso
    - tali aggiornamenti, ovviamente, modificano il sistema operativo in una qualche parte
  - è invece abbastanza frequente che un utente crei nuovi programmi applicativi, per sé o anche per altri
    - per farlo, occorre conoscere un qualche *linguaggio di programmazione*, ed avere degli opportuni programmi applicativi con la seguente caratteristica: si tratta di programmi (che si chiamano *compilatori*) che creano altri programmi
    - in alcuni casi, anziché di compilatori, si parla di *interpreti*, non trattati in questo corso
    - un po' a parte è il discorso sugli applicativi creati per gestire basi di dati con DBMS; se ne tratterà a fine corso

- tipicamente, si acquista il sistema operativo insieme col computer (sistema operativo *preinstallato*)
  - nel caso dei Mac di Apple, praticamente ci può essere solo una delle versioni di MacOS-X
  - nel caso degli IBM-compatibili, nel 99% dei casi (fino a 2 anni fa quasi il 100%) c'è Windows

  - al giorno d'oggi ci sono anche computer con Linux come sistema operativo
    * costano una 50ina di euro in meno, perché Linux è gratuito
  - si possono acquistare IBM-compatibili anche senza sistema operativo
  - in questo caso, una volta che lo si accende, lui fa qualche controllo e poi si blocca con una schermata nera lamentandosi di non avere un sistema operativo

- di solito, l'utente normale si limita ad *aggiornare* il sistema operativo
  - di solito per correggere qualche problema di sicurezza riscontrato di recente
  - anche cambiare il sistema operativo alla versione successiva (*upgrade*, ad esempio da Windows XP a Windows Vista) è sostanzialmente un aggiornamento
  - qualche volta capita di dover riavviare il computer, ma è comunque una cosa semplice da fare

- tuttavia, è possibile anche *disinstallare* il sistema operativo (pre)installato e installarne un altro
  - operazione non semplicissima, ma neanche impossibile
  - con i Mac, lo fanno solo i superhacker
  - sugli IBM-compatibili ci si può sbizzarrire a volontà
  - la cosa tipica che fa chi si intende un po' di informatica è fare un sistema *dual-boot*
    * quando lo si accende, viene chiesto se far partire Windows o Linux
    * con il termine "boot" si intende quello che fa il computer non appena lo si accende
    * in pratica è la fase in cui viene "caricato" il sistema operativo
    * ovvero, la fase il sistema operativo fa alcune cose iniziali che gli permetteranno poi di funzionare
    * può durare da pochi secondi e un paio di minuti
  - aggiornare un sistema operativo (passare da una versione precedente ad una più recente) è facile
  - cambiare totalmente un sistema operativo (da Windows a Linux o viceversa) è più complicato
    * se non si sta attenti, si perdono tutti i dati

- l'accensione di un computer: cosa succede?

1. *bootstrap* (o solo *boot*): una parte di software indipendente dal sistema operativo fa alcuni controlli sul funzionamento di alcune parti dell'hardware
   - ad esempio, vede quanta RAM c'è e controlla che funzionino tastiera e monitor
   - questa parte di software si trova su una memoria RAM che però non è volatile, e neanche modificabile
   - per questo motivo viene chiamata ROM (Read Only Memory)
   - nei sistemi IBM-compatibili, questa procedura software si chiama BIOS (Basic Input Output System)
2. l'ultima parte del boot fa partire il sistema operativo, da disco, da CD o da floppy
   - è possibile scegliere la *sequenza di boot* all'inizio
   - infatti, alcune parti del BIOS sono modificabili
   - in pratica, occorre premere un qualche tasto (ad esempio F9) nei primi 2-3 secondi di accensione del computer
3. dopodiché le istruzioni da eseguire vengono tratte dal disco, da CD o da floppy a seconda della scelta precedente
   - il sistema operativo vero e proprio parte praticamente sempre da disco
   - negli altri casi:
     * si è su un sistema piccolo o senza hard disk, e quindi il dischetto o il CD contengono il sistema operativo
     * oppure si vuole installare un nuovo sistema operativo che si trova su dischetto (difficile) o su CD
     * in questo caso, si verrà avviati ad una serie di passi che costituiscono il programma d'installazione
     * oppure è successo qualcosa di grave e il dischetto o il CD contengono programmi di recovery
4. seguendo tali istruzioni (supponendo che non si tratti di installazione o di recovery), il sistema operativo comincia pian piano a posizionare le sue parti vitali in RAM
5. quando è tutto pronto, si arriva alla schermata di login
   - occorre immettere username e password
   - sui PC (soprattutto Windows), questa parte viene spesso saltata
   - in pratica, si accede come utente generico (*user* o *guest*) senza password
   - sui PC non personali (quelli dei laboratori), chi amministra il sistema (dei tecnici pagati per questo) deve aver rilasciato all'utente un *account*, ovvero uno username e una password

- di solito, viene chiesto di modificare subito la password al primo accesso
- a meno che non si sia amici dell'amministratore, solitamente non è possibile cambiare lo username
- c'è sempre uno user onnipotente (solitamente chiamato *superuser*, *root* o *administrator*)
- è l'unico che può modificare il sistema operativo, installando applicazioni o facendo aggiornamenti

6. poi viene caricata la GUI (Graphical User Interface) e si può cominciare ad usare il computer
   - alcuni vecchi sistemi operativi non avevano la GUI, oppure occorreva farla partire come un'applicazione in un secondo momento
   - in questi casi, c'era una schermata nera con un *prompt* che attendeva comandi da tastiera
   - oggi la situazione è invertita: si parte con la GUI ed eventualmente si fa partire la schermata nera come un'applicazione
7. quando si finisce si fa *logout* (ci si *disconnette*) ed eventualmente si spegne il computer
   - fare logout senza spegnere equivale a permettere ad altre persone con un altro account di accedere
   - lo spegnimento va fatto da software, chiedendo al sistema operativo di spegnersi
   - infatti, per potersi poi riavviare correttamente la prossima volta, occorre che salvi alcune informazioni prima di spegnersi
   - per questo motivo, se si preme normalmente sul pulsante di accensione, il computer resta accesso
   - se si vuole proprio spegnere senza permettere al sistema operativo i suoi salvataggi usuali (cosa ad esempio necessaria se il sistema operativo stesso si è bloccato) occorre tener premuto il tasto di spegnimento per qualche secondo

- per usare il sistema operativo ci sono sostanzialmente 2 modi
  - da interfaccia grafica (quindi con la GUI)
    - si va avanti a clic, doppi clic e trascinamenti (drag and drop) col mouse
    - la tastiera viene usata solo quando è indispensabile
      - per esempio, per scrivere una lettera con Word, prima si apre Word e poi si scrive la lettera
      - per andare su un sito, prima si apre il Browser (es. Internet Explorer o Firefox) e poi si scrive l'indirizzo
    - ci sono svariati modi per eseguire un programma

        - fare doppio clic su un'icona sul Desktop
        - fare clic in una barra di avviamento veloce
        - selezionare il programma dai menu
      * all'interno di un programma, si può usare il mouse o la tastiera in tanti modi, dipendentemente dal programma stesso
        - ad esempio, sul browser si usa prevalentemente il mouse per cliccare sui link, su Word si usa prevalentemente la tastiera per scrivere
  - da interfaccia a linea di comando (CLI, Command Line Interface)
    * è più da geek
    * l'utente "normale" non lo fa mai
    * in maniera pura e "totale" (ovvero usando un sistema operativo che, una volta avviato, mostra solamente lo schermo nero dei comandi), non lo si fa più
    * quello che talvolta qualche utente (non proprio "normale") fa è di eseguire un programma applicativo che apre uno schermetto nero, dove si possono dare comandi direttamente al sistema operativo
    * in Windows, questa applicazione si chiama `Prompt dei comandi` o `DOS prompt`
    * sotto Linux la cosa è più naturale, e ci sono svariate applicazioni: `konsole` e `gnome-terminal` sono le più comuni
    * i tipici comandi che si danno riguardano il file-system, o servono per lanciare altri programmi
      - è un po' più potente della GUI, perché si possono lanciare i programmi con i *parametri*
      - non che con la GUI non si possa fare, ma talvolta non è altrettanto agevole
    * con la GUI, si fanno le stesse cose ma per quanto riguarda il filesystem occorre prima lanciare qualche altra applicazione (es. Explorer)

- tutti i moderni sistemi operativi (almeno quelli che interessano a noi) sono *multitasking*

  1. lo possono usare contemporaneamente più utenti
     - sia in modo lasco, come ad esempio nella seguente situazione: un utente fa login, lancia e lascia un programma in esecuzione e fa logout, e poi fa login un altro utente che lancia altri programmi (mentre il programma del primo è ancora in esecuzione)
     - sia in modo massivo, cioè il computer è connesso in rete e più utenti contemporaneamente ci accedono con programmi appositi

    - era quello che succedeva nei vecchi mainframe con i *terminali*
  2. uno stesso utente può far eseguire più applicazioni contemporaneamente
  3. in entrambi i casi, le varie applicazioni si alternano, sotto il comando del sistema operativo, nell'uso delle risorse della macchina
    - prima fra tutte la CPU...
    - la parte del sistema operativo che si occupa di ciò è chiamata scheduler (che si può tradurre con "organizzatore")
    - se due utenti, o uno stesso utente, lanciano due programmi che richiedono molte risorse di computazione (CPU, RAM o altro), allora si potrà notare un decremento delle prestazioni del computer

- file system
  - tutti i vari dispositivi di memorizzazione di massa (dischi fissi e non, chiavi USB, CD e DVD) sono organizzati con un file system
  - vuol dire che le informazioni sono memorizzate in archivi (*files*), ad ognuno dei quali occorre dare un nome
  - è compito dell'utente dare un nome a ciascun file in maniera da ricordare poi che informazioni ci sono dentro
    - fanno eccezione a questa regola i files di sistema
    - cioè quelli che servono al sistema operativo
    - lo stesso sistema operativo è inizialmente un file, prima del boot
    - i files di sistema non possono essere toccati, a meno di non essere superuser
  - dato che i files sono spesso molti, non è comodo dover dare un nome diverso a ciascuno di essi
  - e comunque sarebbe difficile cercare un files tra migliaia di altri simili
  - quindi i file systems offrono di organizzare il tutto in cartelle (*directories*)
    - sono dei files anche le directories, ma sono files speciali
    - servono per contenere altri files
  - le directory sono organizzate ad albero
    - c'è una directory iniziale che contiene tutte le altre (`/` in Linux, `C:\` in Windows)
    - ogni directory può contenere sia files che altre directories
    - all'interno di ogni directory, non ci possono essere 2 files con lo stesso nome
  - operazioni tipiche (per files si intende sia files regolari che directory)

    - creare un file vuoto (con un nuovo nome)
    - cancellare un file
        - nel caso di una directory si cancella tutto quello che c'è dentro
        - tipicamente, il sistema operativo chiede conferma
    - copiare un file in una diversa directory o nella stessa ma con un altro nome
    - spostare un file in una diversa directory o nella stessa ma con un altro nome
- tipica forma del nome di un file: `qualcosa.estensione`
    - l'estensione è tipicamente di 3 lettere, ma non necessariamente
    - serve per capire con quale applicazione aprire il file relativo