Interfaccia Utente a caratteri (testuale)
Interprete dei comandi
Shell scripting

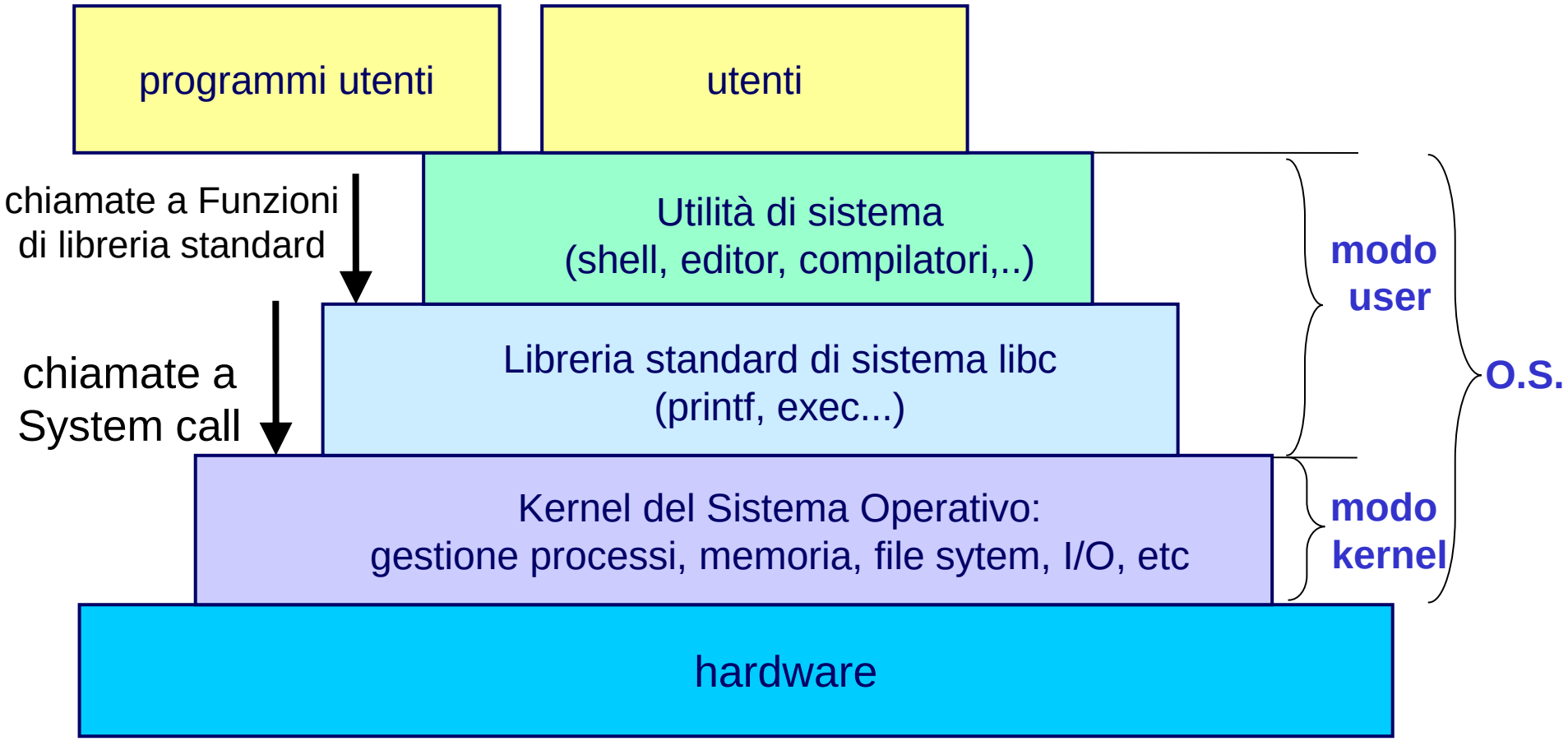
NOTA BENE:

Questa guida contiene solo una introduzione minimale all'utilizzo dell'interprete di comandi bash, e serve per fornire agli studenti del corso di Sistemi Operativi le informazioni iniziali per utilizzare una semplice interfaccia a riga di comando con cui compilare programmi in linguaggio ANSI C mediante il compilatore gcc, eseguire e debuggare tali programmi, scrivere ed eseguire semplici script, capire le interazioni tra i programmi e l'utente e, in sostanza, identificare le funzionalita' messe a disposizione dell'utente dal sistema operativo.

Per tale motivo, alcuni concetti sono stati volutamente semplificati, allo scopo di rendere più semplice la comprensione, anche a discapito della realtà.

Vittorio Ghini

Struttura del Sistema Operativo



Requisito Hardware per il Sistema Operativo
Modi di esecuzione della CPU

Instruction Set Architecture of IA-32

Privilege Levels (Rings)

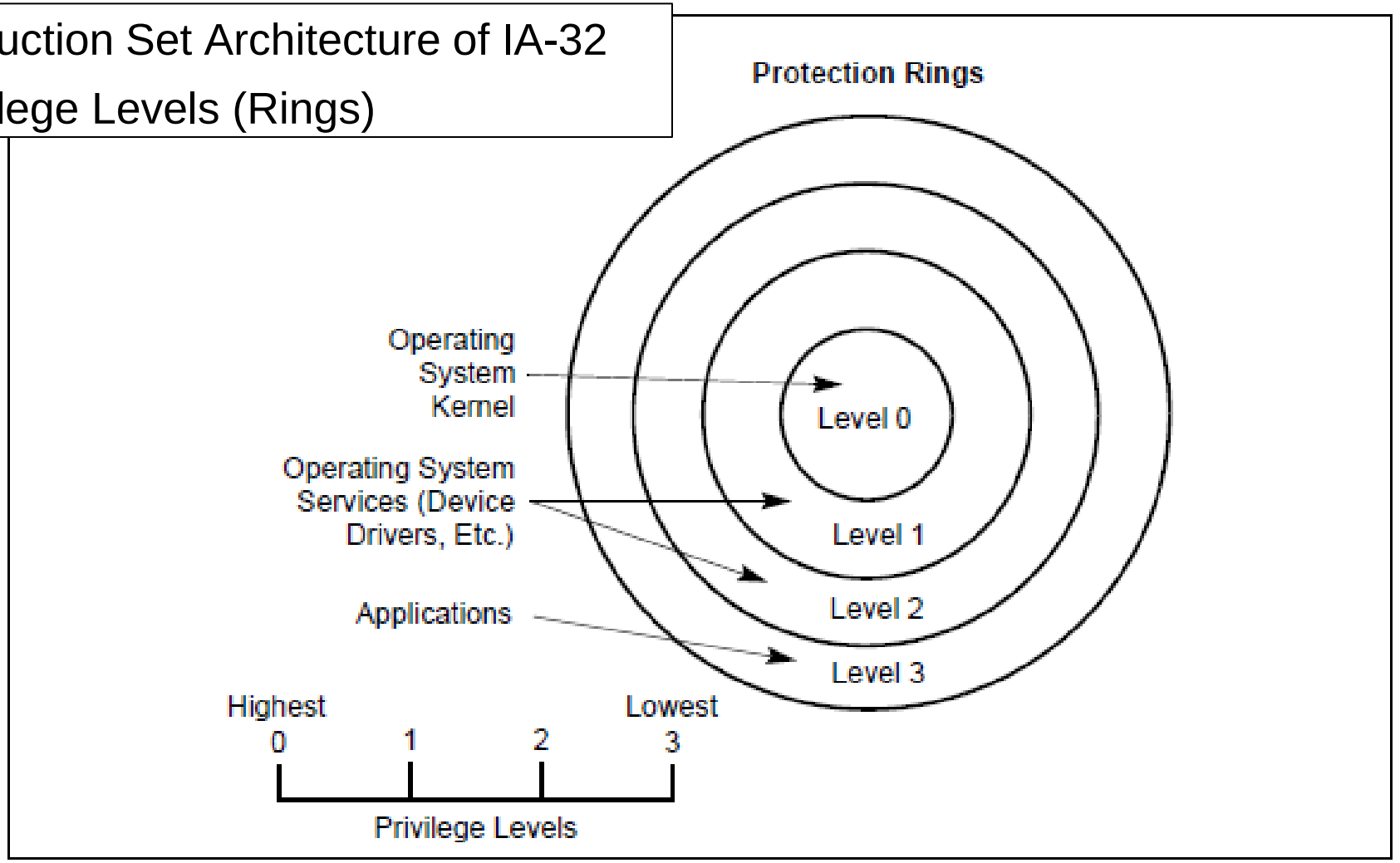


Figure 6-3. Protection Rings

Tratto da : IA-32 Volume 3A System Programming Guide.pdf by Intel

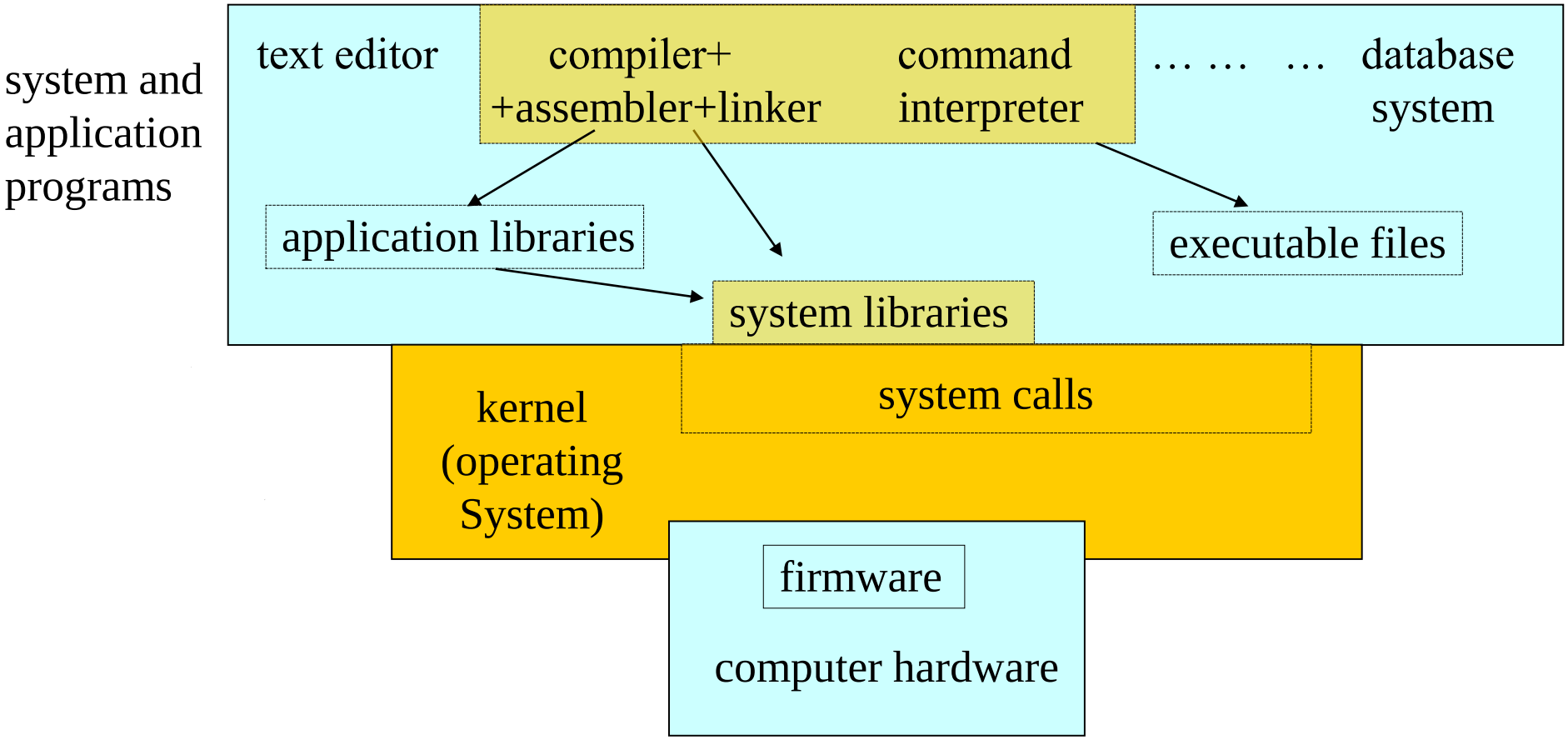
chiamata a system call mediante interrupt

```
/*  file print.s   Descr.: stampa a video di stringa Architettura: x86      Dialetto assembly AT&T    (-masm=att)
 Assemblaggio e Linking      Modo 0, non usare gcc:
       Assemblare con: as -o print.o --gstabs print.s  (eventuale opzione --gstabs per poter usare gdb o ddd)
       Linkare con:     ld -o print.exe print.o
 Eseguire con ./print.exe o (per il debug) con gdb print.exe o ddd print.exe */

.data
miastringa: .string "MANNAG\n" # stringa da stampare compresa di a capo
.text
.globl _start
_start:
 nop
/* qui attivo due volte, mediante l'istruzione int, l'interruzione software identificata dal valore 0x80=8016
con la quale, in generale, si richiamano routine o servizi del sistema operativo (nel nostro caso Linux).
Il servizio richiamato e' identificato dal valore inserito nel registro eax prima di attivare la interruzione.
- Il valore 4 chiede il servizio di stampa di una stringa.
- Il valore 1 invece corrisponde alla routine di terminazione del programma e ritorno al sistema operativo.
*/
/* stampo stringa e vado a capo */
 movl $4, %eax                    /* servizio da ottenere: stampa */
 movl $1, %ebx                    /* sottoservizio da ottenere */
 movl $miastringa, %ecx           /* indirizzo di inizio stringa in registro ecx */
 movl $7, %edx                    /* lunghezza della stringa da stampare in registro edx */
 int $0x80              /* ordine di eseguire interrupt */
/* termino */
fine:
 movl $1, %eax                    /* servizio da ottenere: terminazione processo */
 int $0x80              /* ordine di eseguire interrupt */
```

Librerie e Chiamate di sistema (1)

Ricordiamo l'organizzazione del computer, in particolare la relazione tra le system calls e le librerie di sistema.



Librerie e Chiamate di sistema (2)

- Le chiamate di sistema forniscono ai processi i servizi offerti dal SO:

  Controllo dei processi, Gestione dei file e dei permessi, Gestione dei dispositivi di I/O, Comunicazioni.

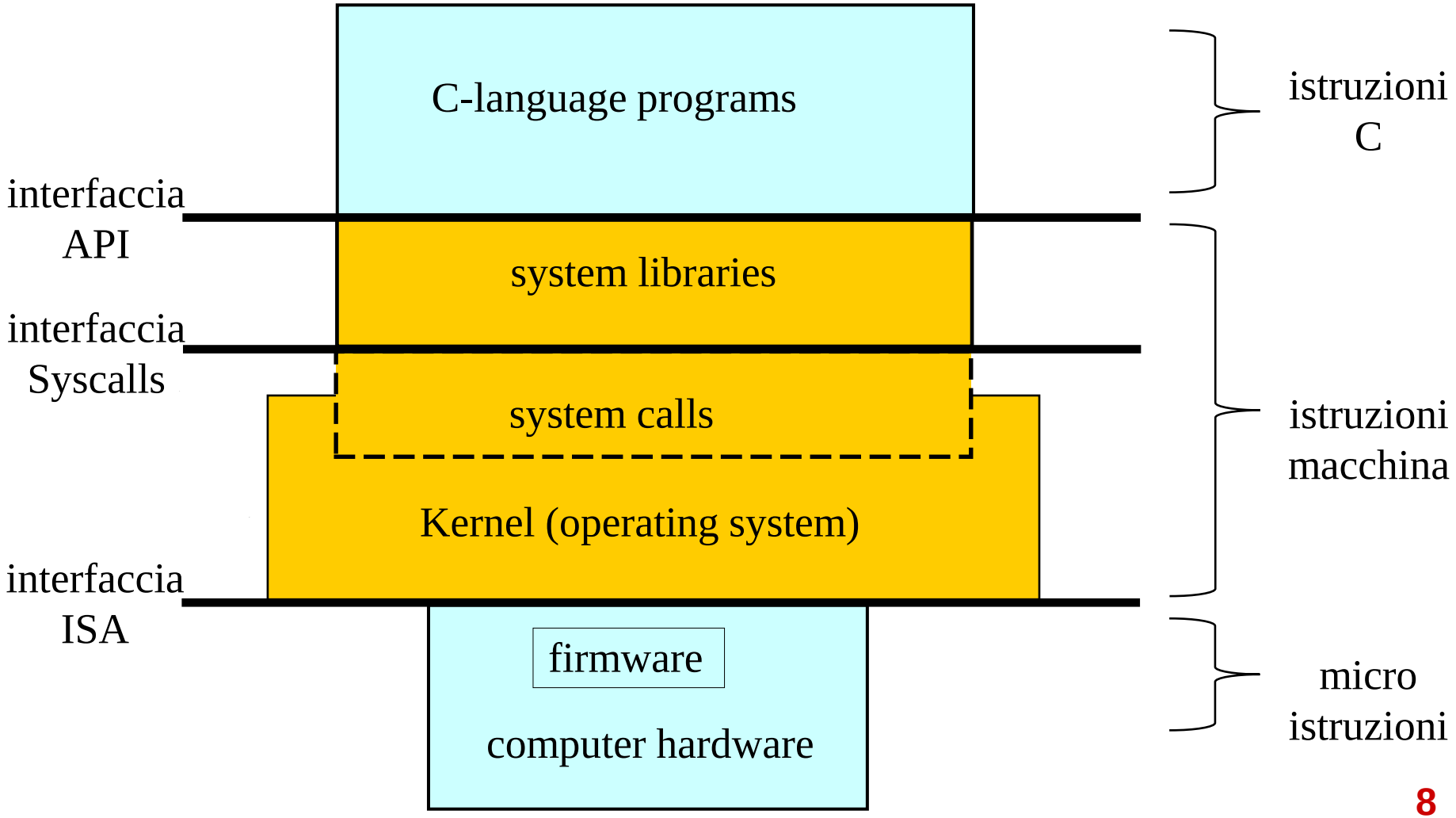
- Il modo in cui sono realizzate cambia al variare della CPU e del sistema operativo.
- Il modo di utilizzarle (chiamarle) cambia al variare della CPU e del sistema operativo.
- Sono utilizzate (invocate) direttamente utilizzando linguaggi di basso livello (assembly).
- Possono essere chiamate (invocate) indirettamente utilizzando linguaggi di alto livello (C o C++)
- Infatti, normalmente i programmi applicativi non invocano direttamente le system call, bensì invocano funzioni contenute in librerie messe a disposizione dal sistema operativo ed utilizzate dai compilatori per generare gli eseguibili. L'insieme di tali funzioni di libreria rappresentano le **API** (**Application Programming Interface**) cioè l'interfaccia che il sistema operativo offre ai programmi di alto livello.
- Le librerie messe a disposizione dal sistema operativo sono dette **librerie di sistema**.

Librerie e Chiamate di sistema (3)

- Per ciascun linguaggio importante, il s.o. fornisce API di sistema da usare.
  - **Cambiando il sistema operativo oppure cambiando il processore (CPU) su cui il sistema operativo si appoggia, i nomi delle funzioni delle API rimangono gli stessi ma cambia il modo di invocare le system call, quindi cambia l'implementazione delle funzioni delle API per adattarle alle system call e all'hardware.**
  - Perché è importante programmare usando linguaggi di alto livello (es C)

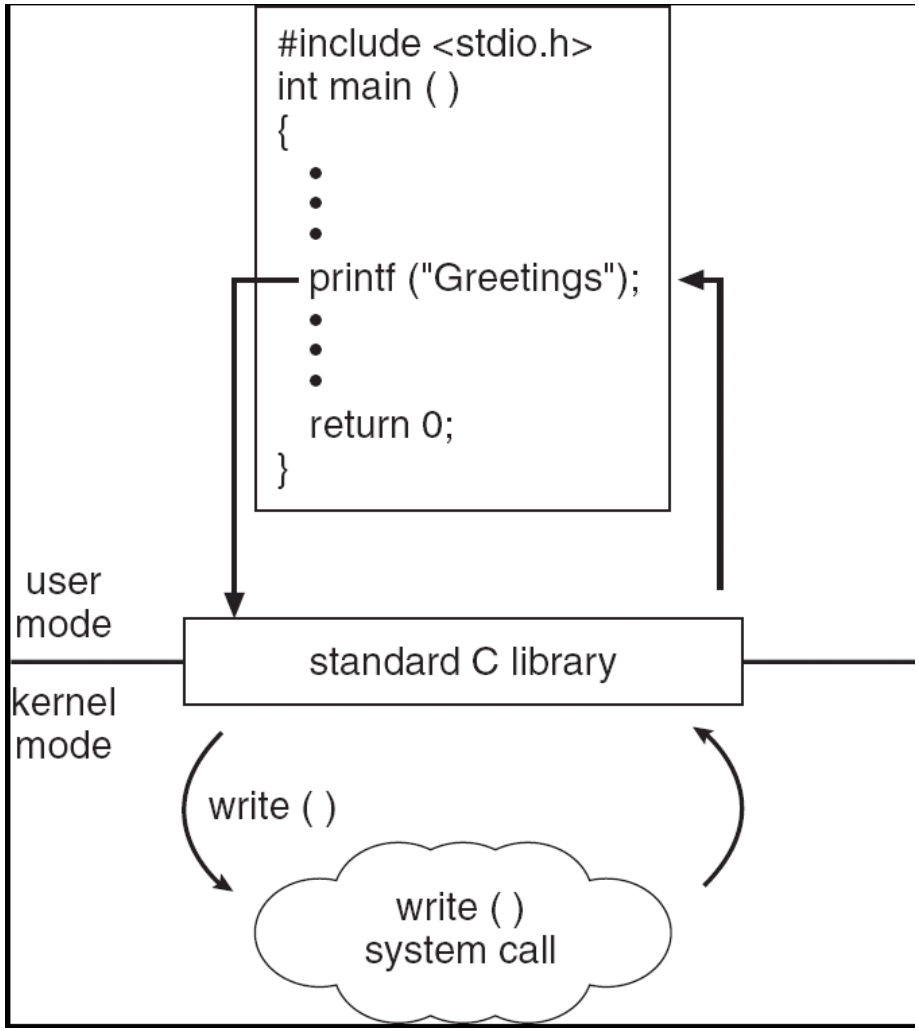
    Un programma sorgente in linguaggio C che usa le API non deve essere modificato se cambio il sistema operativo o l'hardware (e questi mantengono le stesse API), poiché sul nuovo s.o. cambia l'implementazione delle API e quindi cambia l'eseguibile che viene generato su diversi s.o ed hardware.
- Alcune API molto diffuse sono:
  - Win32 API per Windows
  - **POSIX API per sistemi POSIX–based (tutte le versioni di UNIX, Linux, Mac OS X)**
  - Java API per la Java Virtual Machine (JVM).
- Possono esistere anche librerie messe a disposizione non dal sistema operativo bensì realizzate, ad esempio, da un utente.
- Solitamente tali librerie applicative sono implementate utilizzando le librerie di sistema.

Linguaggi utilizzati e interfacce di servizio



Esempio con la libreria standard C

- Per Linux, la libreria standard del linguaggio C (il *run–time support system*) fornisce una parte dell'API



- Programma C che invoca la funzione di libreria per la stampa *printf()*
- La libreria C implementa la funzione e invoca la system call *write()* **nel modo richiesto dallo specifico s.o. e dalla specifica CPU.**
- La libreria riceve il valore restituito dalla chiamata al sistema e lo passa al programma utente

Quali system call utilizza un eseguibile? strace

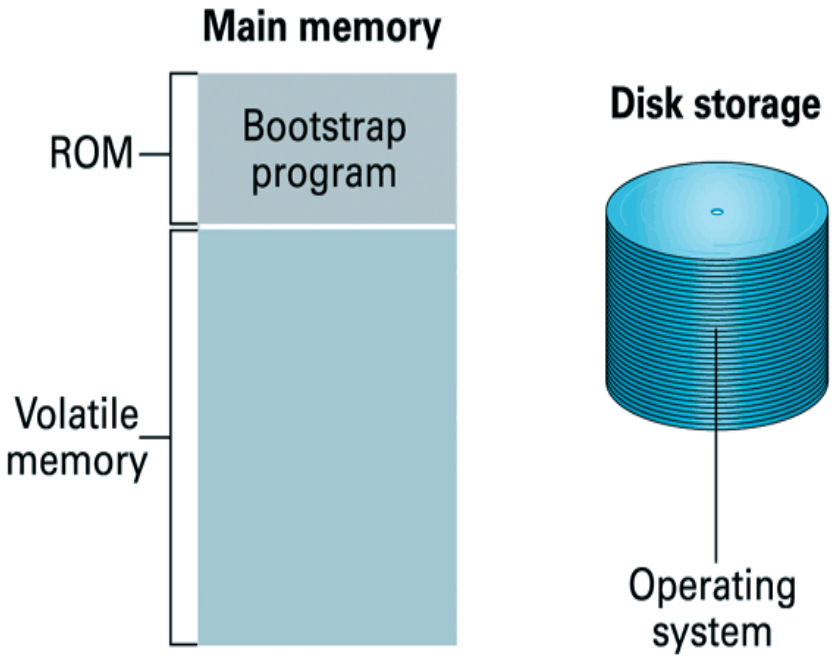
- In Linux esiste un comando che permette di vedere quali system call sono usate da un programma in esecuzione.
- **strace** esegue il comando specificato come argomento fino a che questo termina. Intercetta e visualizza le system calls che sono chiamate dal processo e i segnali che sono ricevuti dal processo.

Facciamo un esempio: lanciando il comando ifconfig visualizzo lo stato delle interfacce di rete. Vediamo quali syscall usa ifconfig eseguendo il comando:

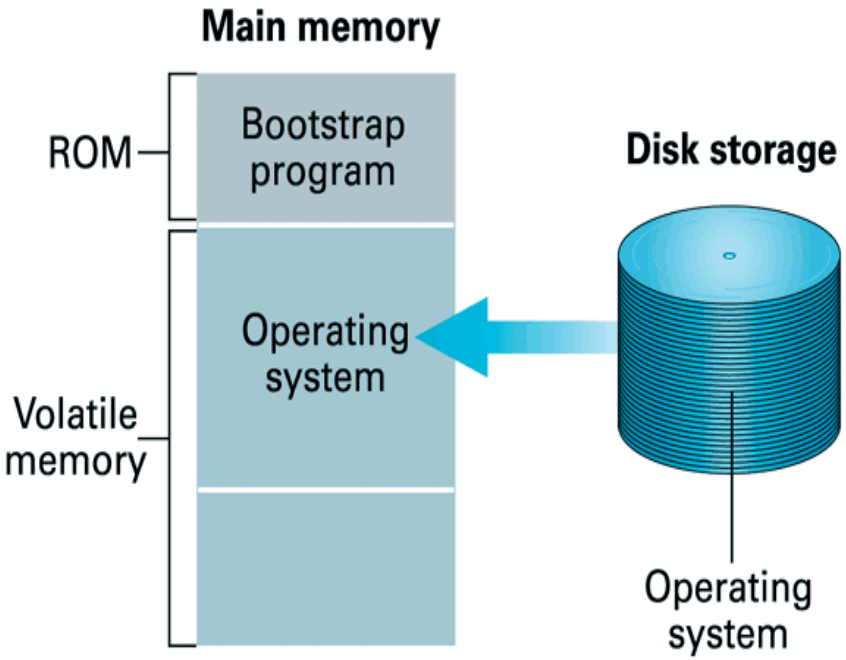
strace ifconfig

```
execve("/sbin/ifconfig", ["ifconfig"], [/* 52 vars */]) = 0
brk(NULL)                               = 0x12f3000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap(NULL,                              8192,                           PROT_READ|PROT_WRITE,
        MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f37cb8ee000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=71665, ...}) = 0
mmap(NULL, 71665, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f37cb8dc000
close(3)
```

L'avvio del sistema operativo - Bootstrap



Step 1: Machine starts by executing the bootstrap program already in memory. Operating system is stored in mass storage.



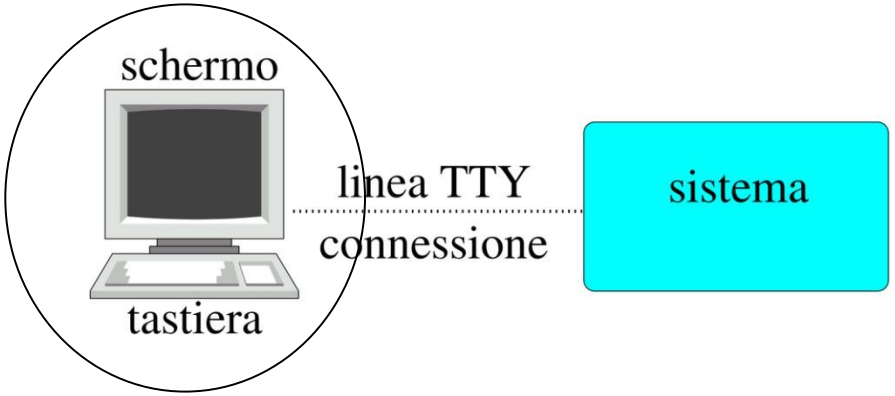
Step 2: Bootstrap program directs the transfer of the operating system into main memory and then transfers control to it.

Servizi del Sistema Operativo

- **Gestione di Risorse**: allocazione, contabilizzazione, protezione e sicurezza (possessori di informazione devono essere garantiti da accessi indesiderati ai propri dati; processi concorrenti non devono interferire fra loro).
- Comunicazioni (intra e inter-computer)
- Rilevamento di errori che possono verificarsi nella CPU e nella memoria, nei dispositivi di I/O o durante l'esecuzione di programmi utente
- Gestione del file system — capacità dei programmi e dell'utente di leggere, scrivere e cancellare file e muoversi nella struttura delle directory
- Operazioni di I/O — il SO fornisce ai programmi utente i mezzi per effettuare l'I/O su file o periferica
- Esecuzione di programmi — capacità di caricare un programma in memoria ed eseguirlo, eventualmente rilevando e gestendo, situazioni di errore
- **Programmi di sistema**
- **Chiamate di sistema**
- **Interfaccia utente**: **a linea di comando** (*Command Line Interface*, CLI) o grafica (*Graphic User Interface*, GUI).

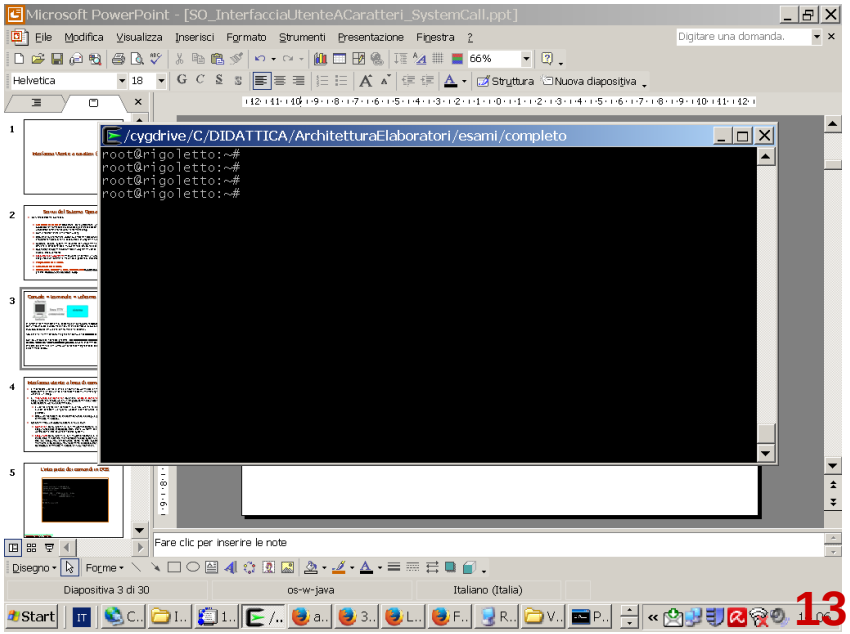
Console = terminale = schermo + tastiera

In origine i terminali (schermo+tastiera) erano dispositivi **separati** dal computer vero e proprio (detto mainframe) e comunicavano con questo mediante una linea seriale. L'output verso l'utente era di tipo solo testuale, l'input era fornito mediante tastiera.



Attualmente i terminali sono integrati nei computer (**schermo e tastiera del PC**).

Con l'avvento delle interfacce grafiche, **lo schermo del terminale viene emulato in una "finestra" dell'ambiente grafico**. Si parla di terminale virtuale. Quando la finestra del terminale è in primo piano i caratteri digitati dalla tastiera vengono passati al terminale stesso.



Nozioni per l'uso del Terminale: File system (1)

Il filesystem è la organizzazione del disco rigido che permette di contenere i file e le loro informazioni.

Lo spazio di ciascun disco rigido è suddiviso in una o più parti dette partizioni.
Le partizioni contengono
- dei contenitori di dati detti files
- dei contenitori di files, detti directories (folders o cartelle in ambienti grafici)..

In realtà ciascuna directory può a sua volta contenere dei files e anche delle altre directories formando una struttura gerarchica

In Windows le partizioni del disco sono viste come logicamente separate e ciascuna e' indicata da una lettera (C; B: Z: …).

Se un file si chiama pippo.c ed è contenuto in una directory che si chiama vittorio che a sua volta è contenuta in una directory che si chiama home che a sua volta è contenuta nella partizione chiamata C:, allora è possibile individuare univocamente il file pippo.c indicando il **percorso** mediante il quale, partendo dalla partizione C: si arriva al file pippo.c

C:\home\vittorio\pippo.c <- percorso per raggiungere pippo.c

Notare il carattere separatore **** (si chiama backslash) che indica dove inizia il nome di una directory o di un file.

Nozioni per l'uso del Terminale: File system (2)

In Linux/Mac invece le partizioni sono viste come collegate tra loro.
Esiste una partizione principale il cui nome è ***/*** (slash).
Questa partizione, così come le altre partizioni, può contenere files e directories. Le directoryes possono contenere files e directories.

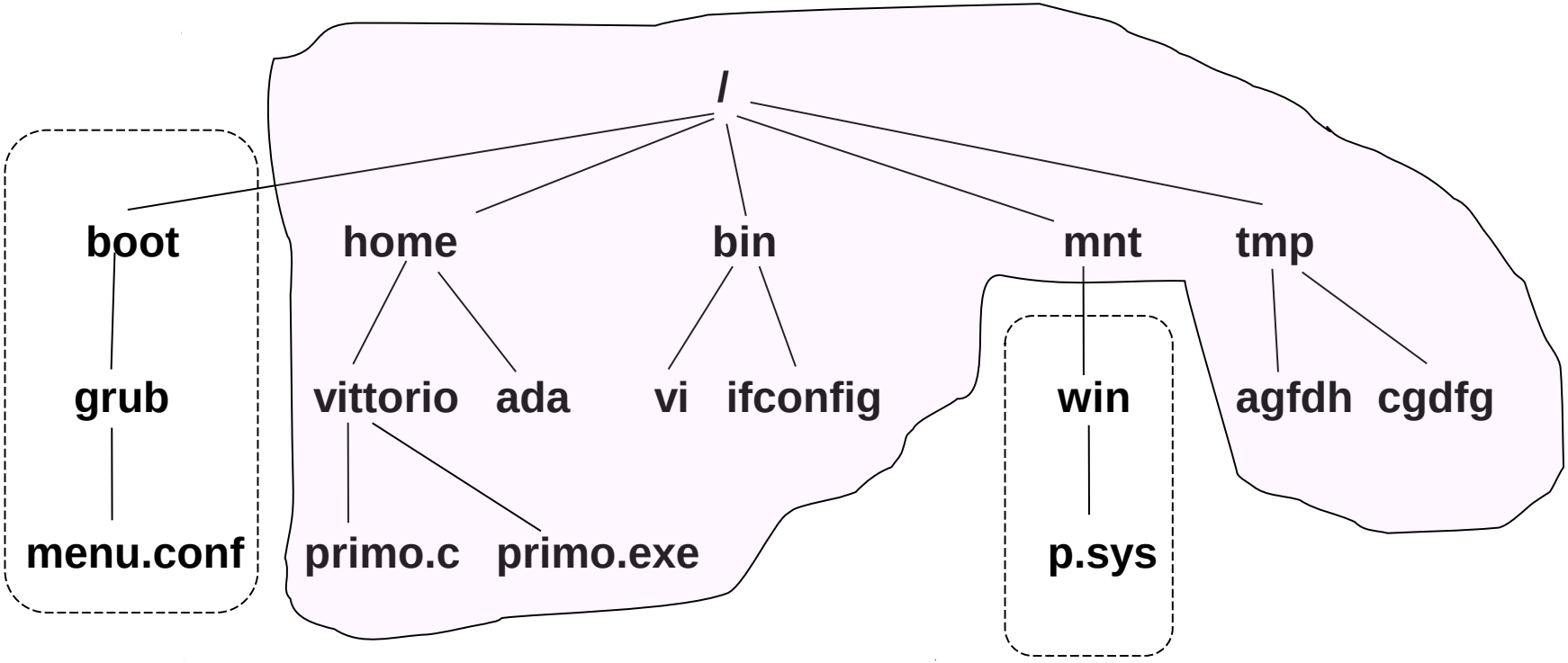
Se un file si chiama primo.c ed è contenuto in una directory che si chiama vittorio che a sua volta è contenuta in una directory che si chiama home che a sua volta è contenuta nella partizione principale, allora è possibile individuare univocamente il file pippo.c indicando il **percorso** mediante il quale, partendo dalla partizione / si arriva al file primo.c
/home/vittorio/primo.c <- percorso per raggiungere primo.c

Notare il carattere separatore ***/*** (slash) che indica dove inizia il nome di una directory o di un file. Quando il separatore / è all'inizio del percorso invece indica l'inizio della partizione principale, inizio che viene detto **root.**.

Le partizioni diverse da quella principale, si innestano (si collegano) logicamente in una qualche directory della partizione principale..
Ad esembio, una partizione chiamata boot può innestarsi direttamente nell'origine della partizione principale. Se questa partizione contiene una directory grub e quella directory contiene un file menu.conf, allora il percorso per identificare quel file menu.conf sarà:
/boot/grub/menu.conf <- percorso per raggiungere il file.conf

Notare che non si capisce se un nome indica una directory o una partizione.

Nozioni per l'uso del Terminale: File system (3)



Esempio di strutturazione in directories e files delle partizioni di un filesystem Linux:

Nell'esempio, alla partizione principale ***/*** sono collegate (tecnicamente si dice **montate**) altre due partizioni (circondate dalle linee tratteggiate) di nome **boot** e **win**..

Notate che mentre la partizione boot è montata direttamente come fosse una directory contenuta nella directory di inizio (detta root) della partizione principale (/), la partizione win è montata in una sottodirectory della root.

Nozioni per l'uso del Terminale: File system (4)

Nei sistemi **Linux/Mac** ciascun utente di un computer ha a disposizione dello spazio su disco per contenere i suoi files.

Per ciascun utente esiste, di solito, una directory in cui sono contenuti i files (intesi come directories e files) di quell'utente.
Quella directory viene detta **home dell'utente**.

Per esempio, nei pc dei laboratori del corso di laurea in Informatica di Bologna, la home dell'utente "rossi" si trova in **/home/students/rossi**

Per accedere ad un computer ciascun utente deve utilizzare farsi riconoscere mediante un nome utente (**account**) e autenticarsi mediante una **password**.
L'operazione iniziale con cui l'utente si autentica per accedere ad un pc si dice login.

L'autenticazione degli utenti permette che il sistema operativo protegga i files di un utente impedendo l'accesso da parte di altri utenti o di persone esterne.
Ciascun utente può vedere quali sono i permessi di accesso ai propri files ed eventualmente modificarli.

Nozioni per l'uso del Terminale: File system (5)

Nel momento in cui si accede al terminale a riga di comando di un computer, il terminale **stabilisce la posizione logica attuale dell'utente all'interno del filesystem**, collocandolo inizialmente nella propria home directory.

Durante il lavoro l'utente può spostare la propria **posizione logica** in una diversa directory del filesystem.

NOTA BENE: spostarsi logicamente in una diversa directory VUOL DIRE VISITARE quella diversa directory e NON VUOL DIRE TRASFERIRE I PROPRI FILES IN QUELLA DIRECTORY

La directory in cui l'utente si trova logicamente in questo momento viene detta **directory corrente** e si dice che l'utente "**si trova nella**" directory corrente.

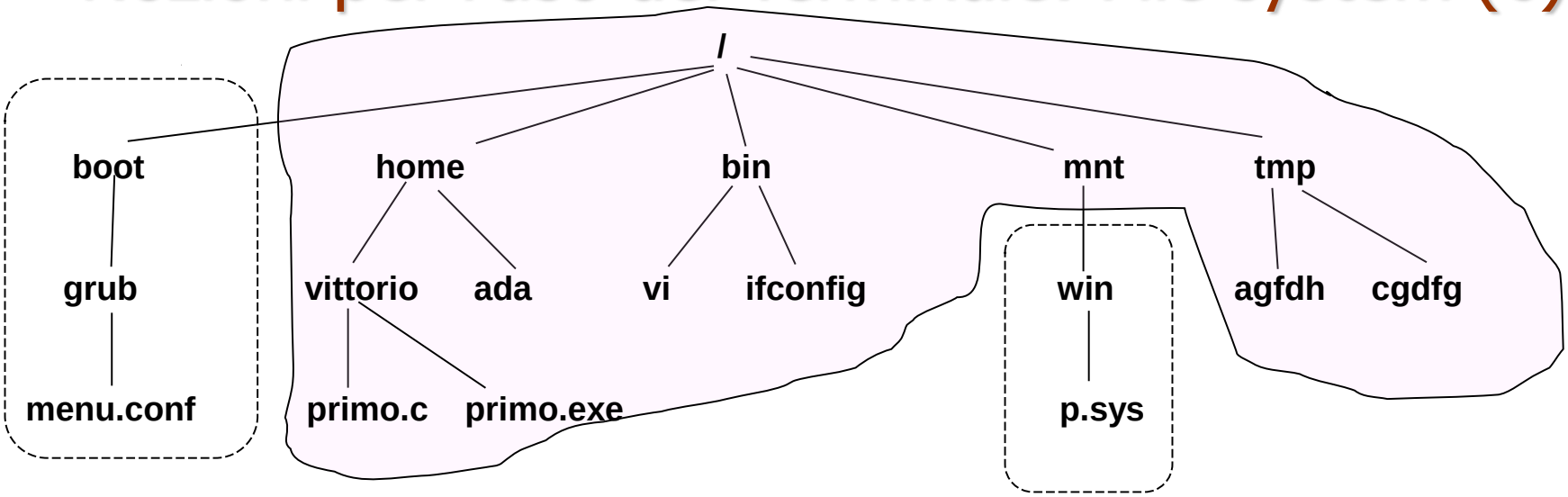
L'utente, per sapere in quale directory si trova logicamente in questo momento può eseguire il comando **pwd**, digitandolo da tastiera per farlo eseguire alla shell. Il comando pwd visualizza sullo schermo il percorso completo da / fino alla directory in cui l'utente si trova in quel momento.

```
 #  pwd
/home/students/rossi/merda
```

Per spostarsi logicamente in una diversa directory, l'utente usa il comando **cd**

```
# cd   /var/log
```

Nozioni per l'uso del Terminale: File system (6)



Supponiamo di trovarci logicamente nella directory /home/vittorio
Per spostarmi logicamente nella directory /home posso usare cd in tre modi diversi

cd /home <- specifico percorso assoluto

cd **../** <- specifico percorso relativo cioè partendo dalla directory corrente

Si noti che **il simbolo .. indica la directory superiore**

Per spostarmi dalla directory /home/vittorio alla directory /mnt/win

cd /mnt/win <- specifico percorso assoluto

cd **../../**mnt/win <- specifico percorso relativo

Per spostarmi dalla directory /boot alla directory /boot/grub

cd /boot/grub <- specifico il percorso assoluto

cd **./**grub <- specifico percorso relativo (**il simbolo . è la directory corrente**)

cd grub <- specifico percorso relativo (più semplice)

FHS – Filesystem Hierarchy Standard

Attualmente versione 3.0.
Specifiche disponibili presso Linux Foundation.
http://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf

- Obiettivo: sapere dove siano posizionati files/directory e quale sia il loro utilizzo
- Vengono specificate un numero minimo di directory
- Per ciascuna Purpose, Requirements, Specific requirements

Directory richieste in ***/***

bin	Essential command binaries
boot	Static files of the boot loader
dev	Device files
etc	Host-specific system configuration
lib	Essential shared libraries and kernel modules
media	Mount point for removeable media
mnt	Mount point for mounting a filesystem temporarily
opt	Add-on application software packages
sbin	Essential system binaries
srv	Data for services provided by this system
tmp	Temporary files
usr	Secondary hierarchy

Utenti e Gruppi

Nei sistemi Unix/Linux esistono le astrazioni di utente (**user**) e gruppo di utenti (**group**).

- Un utente può corrispondere ad una persona umana oppure essere solo la rappresentazione di una entità usata per indicare chi è che esegue un servizio di sistema. Ad esempio lo user mysql che esegue il servizio del database mysql.
- Un utente (**user**) è caratterizzato da una stringa chiamata **username** che contiene il nome utente (studente, vic, syslog) e da un identificatore numerico chiamato **userID** entrambi univoci nel sistema.
- Un gruppo (**group**) è caratterizzato da una stringa chiamata **groupname** che contiene il nome del gruppo (staff, admin,) e da un identificatore numerico chiamato **groupID** entrambi univoci nel sistema. Ciascun utente appartiene ad uno o più gruppi.
- Ciascun file (e ciascuna directory) del filesystem appartiene ad un utente (detto proprietario del file,owner) che normalmente è l'utente che ha creato quel file.
- Ciascun file (e ciascuna directory) è associata ad un gruppo.
- **Un utente può tentare di accedere ad un file**, chiedendo di leggere o modificare il contenuto di un file oppure di eseguire un file eseguibile, **anche se non è il proprietario del file**.
- Viene indicato col termine **effective user** un utente quando cerca di accedere ad un file: il termine permette di distinguere tra chi sta usando il file e chi ne è il proprietario.
- **Il proprietario di un file stabilisce chi può accedere a quel suo file**, configurando i permessi di accesso a quel file. **Il proprietario stabilisce**, distinguendoli, **i permessi assegnati al proprietario del file** (sé stesso), agli utenti appartenenti allo stesso gruppo del file e, infine, **a tutti gli altri**.

Permessi di file e directory

Ogni file ha un **proprietario** (identificato da un numero intero univoco detto **userID**) ed un **gruppo** del proprietario (identificato da un intero detto **groupID**). Allo userID corrisponde una stringa username, e al groupID corrisponde una stringa groupname. Quando un utente crea un file, il s.o. assegna l'utente come proprietario del file appena creato. Il proprietario/creatore poi può cambiare il proprietario del file con il comando

chown *nuovoproprietario nomefile*

Ciascun file mantiene diversi diritti di lettura, scrittura ed esecuzione assegnati al proprietario, al gruppo del proprietario e a tutti gli altri.

Lettura (valore **4**, simbolo **r**)

File: lettura

Directory: elenco file/directory nella cartella (non le proprietà di file e directory)

Scrittura (valore **2**, simbolo **w**)

File: modifica

Directory: creazione, eliminazione, cambio nome file

Esecuzione (valore **1**, simbolo **x**)

File: esecuzione

Directory: accesso all'interno della directory e proprietà di suoi file e directory

user			group			others		
R 4	W 2	X 1	R 4	W 2	X 1	R 4	W 2	X 1

Comandi per cambiare proprietario, gruppo e permessi: ***chown***, ***chgrp***, ***chmod***

Solo il proprietario del file può cambiare proprietario, gruppo e permessi del file.

Permessi di file e directory

Se un utente (detto user ID effettivo) vuole accedere ad un file, si applicano i seguenti criteri di utilizzo basati sui permessi di quel file

- Se l'utente (user ID effettivo) che vuole accedere ad un file è il proprietario del file, si applicano le User permission.
- Altrimenti, se il group ID effettivo corrisponde al group ID del file, si applicano le Group permission
- Altrimenti, si applicano le Other permission

user			group			others		
R	W	X	R	W	X	R	W	X
4	2	1	4	2	1	4	2	1

Solo il proprietario del file può cambiare proprietario, gruppo e permessi del proprio file.
Comandi per cambiare proprietario, gruppo e permessi: *chown, chgrp, chmod*

Esempio di assegnazione contemporanea di permessi **mediante formato numerico**:
assegnazione contemporanea di permessi
per owner (lettura, scrittura e esecuzione: 7),
per group (lettura e scrittura: 6)
e per other (sola lettura: 4)
chmod 764 ./miofile.txt

Visualizzazione Permessi di file e directory

Lanciamo il comando ***ls -al*** nella nostra home directory per vedere tutte le informazioni (opzione -l) e quindi anche i permessi, di tutti i file (opzione -a).

ls -alh /home/vic/

Otteniamo come output:

```
drwxr-xr-x  34 vic  vic   4096 set 25 10:55  .
drwxr-xr-x   3 root root  4096 dic  9  2015  ..
-rw-r--r--   1 vic  vic   3826 giu 16 16:21  .bashrc
-rw-rw-r--   1 vic  vic    158 dic 17  2015  main.c
-rwxrwxr-x   1 vic  vic   8608 dic 17  2015  main.exe
```

Guardiamo i permessi dell'eseguibile che permette di cambiare la propria password.

ls -alh /usr/bin/passwd

Otteniamo come output:

-rw**s**r-xr-x 1 root root 51K lug 22 2015 /usr/bin/passwd

Notare la s **nella parte di permessi utente**, che sostituisce la x di esecuzione.

Dice che quando quell'eseguibile viene eseguito da qualcuno che puo' eseguirlo, il processo creato dall'eseguibile esegue con i permessi di chi lo ha lanciato, **ma anche con i permessi del proprietario dell'eseguibile** (root, nel nostro esempio). Serve per effettuare operazioni con i permessi dell'amministratore di sistema.

Special Permissions		
setuid 4	setgid 2	sticky 1

Permessi speciali di file e directory

setuid - rappresentato da “s” (o da "S") nelle user permissions (settato s con chmod 4***)

File: in esecuzione, il processo associato all'esecuzione del file ottiene anche i diritti dell'owner (l'effective uid diventa quello dell'owner del file).
Tipicamente root. Esempio del comando /usr/bin/passwd

Directory: ignorato

esempio di settaggio setuid per proprietario: chmod u+**s** ./miofile

altro esempio di settaggio numerico di setuid e altri permessi: il 4 all'inizio e' setuid

```
vic@vic:~$ chmod 4761 ./main.exe
vic@vic:~$ ls -alh main.exe
-rwsrw---x 1 vic vic 8,5K dic 17  2015 main.exe
```

setgid - rappresentato da "s" (o da “S”) nelle group permissions (settato con chmod 2***)

File: analogo al setuid ma per il gruppo (è l'effective gid che diventa quello del file)

Directory: implica che i nuovi file e subdirectory create all'interno della directory ereditino il gid della directory stessa (e non quello del gruppo principale dell'utente che lo ha creato). Esempio di una directory condivisa

sticky bit - rappresentato da “t” (o da "T") (settato con chmod 1***)

File: ora ignorato

Directory: i file all'interno di una directory con sticky bit possono essere rinominati o cancellati solo dal proprietario del file, dal proprietario della directory...o da root, ovviamente!