I Sistemi Operativi

Introduzione ai Sistemi Operativi (parte A)

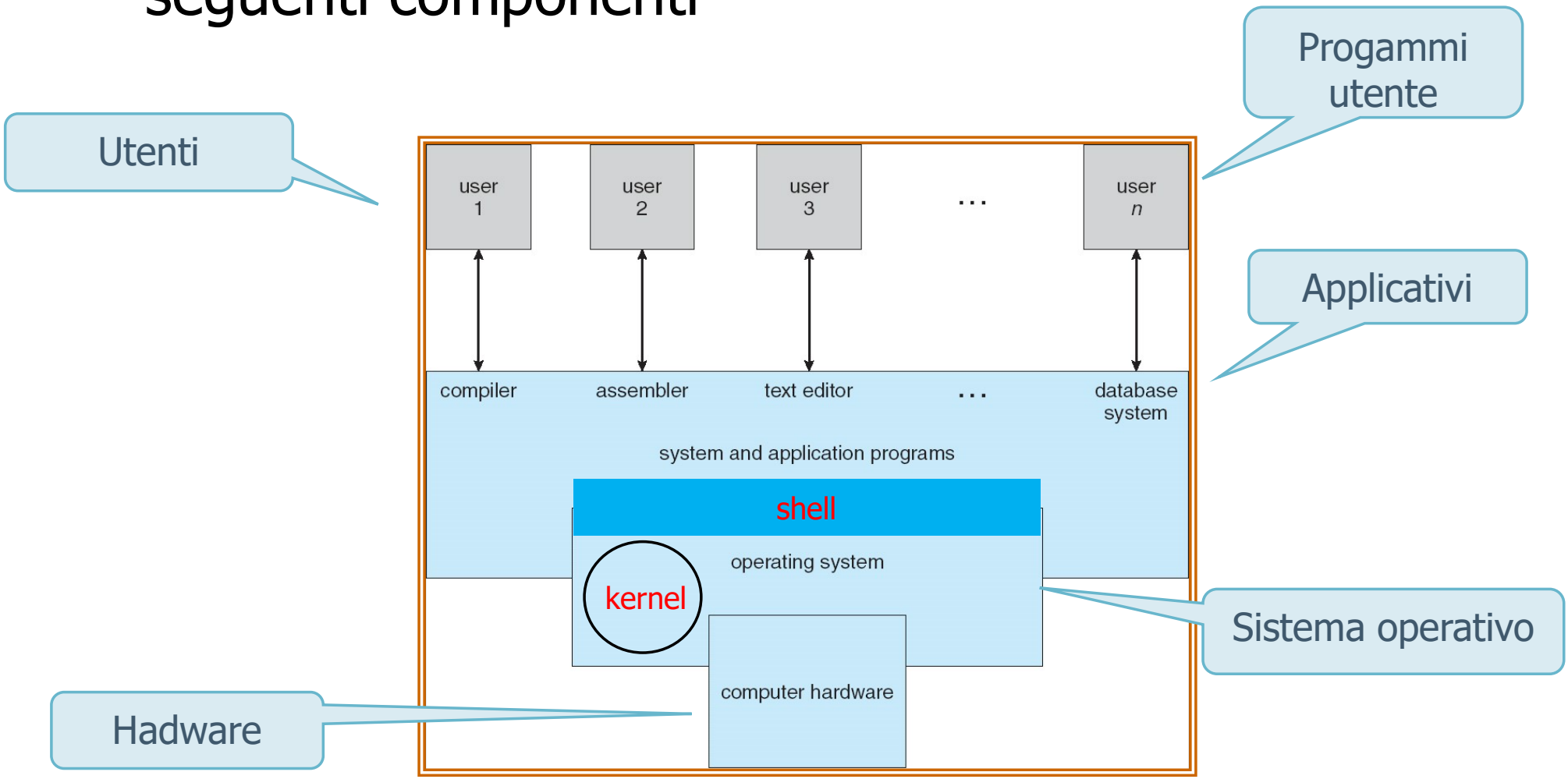
Stefano Quer
Dipartimento di Automatica e Informatica
Politecnico di Torino

Sistema di elaborazione

- Un sistema di elaborazione è costituito dai seguenti componenti
  - Hardware
    - Fornisce le risorse di elaborazione (CPU, memoria, periferiche di I/O, etc.)
  - Sistema operativo
    - Controlla e coordina l'uso dell'hardware
  - Programmi applicativi e di sistema
    - Forniscono i servizi utente (compilatori, database, programmi per office automation, giochi, etc.)
  - Utenti
    - Costituiti da persone, macchine, altri computer

Sistema di elaborazione

- Un sistema di elaborazione è costituito dai seguenti componenti



Sistema operativo ... Hug?!

- Non esiste definizione universale
  - Software eseguito in modalità protetta (o kernel)
- Obiettivi
  - Eseguire comandi e programmi (rendere più facile la soluzione di problemi)
  - Rendere il sistema più facile da utilizzare
  - Usare l'hardware in modo efficiente
- Può essere classificato mediante una visione
  - Top-down
  - Bottom-up

SO: Macchina Virtuale o Estesa

- Una visione top-down di un SO permette di considerarlo come **macchina estesa**
  - Astrae i vari dispositivi del sistema
    - Definisce e implementa le astrazioni
      - I dischi sono visti attraverso i driver e manipolati attraverso il relativo filesystem
    - Utilizza tali astrazioni per risolvere i problemi
  - Attraverso il meccanismo di astrazione
    - Nasconde informazioni e risorse
    - Permette la gestione delle risorse in maniera semplificata

SO: Gestore delle risorse

- Una visione bottom-up di un SO permette di considerarlo **gestore delle risorse**
  - Il SO è un programma che controlla
    - Il funzionamento e le operazioni dei dispositivi
    - L'esecuzione dei programmi utente
  - Può essere considerato come un insieme di **moduli**, ognuno dei quali fornisce determinati **servizi** all'utente

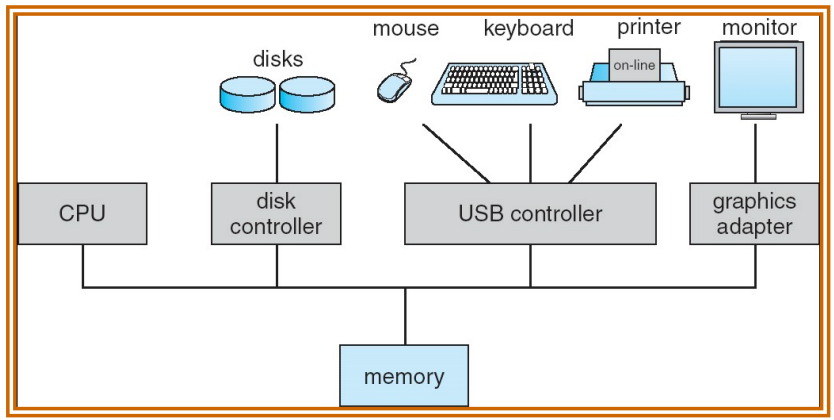
Moduli e Servizi

- Moduli e servizi tipici di un SO
  1. Interprete dei comandi
  2. Gestione dei processi
  3. Gestione della memoria principale
  4. Gestione della memoria secondaria
  5. Gestione dei dispositivi di I/O
  6. Gestione file e file system
  7. Implementazione dei meccanismi di protezione
  8. Gestione delle reti e sistemi distribuiti

Analizzati nel corso

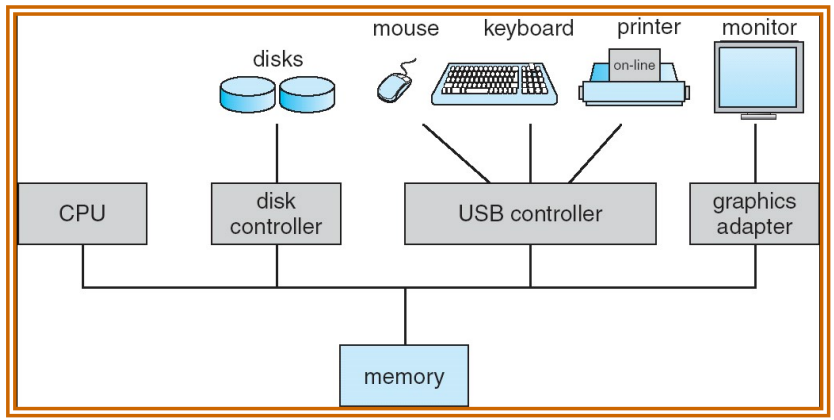
Cenni

Analizzati nel corso



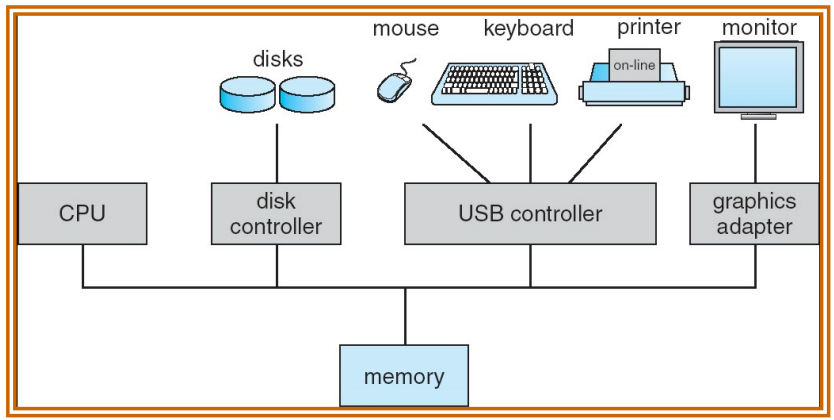
Moduli e Servizi

- Moduli e servizi tipici di un SO
  - Interprete dei comandi
    - L'utente e il SO comunicano attraverso una interfaccia che può essere testuale o grafica
    - L'utente effettua il proprio lavoro mediante un interprete di comandi (shell)
    - Il SO deve permettere all'utente di
      - Gestire i processi
      - Gestire la memoria principale e quella secondaria
      - Instaurare politiche di protezione
      - Gestire la rete e le connessioni esterne



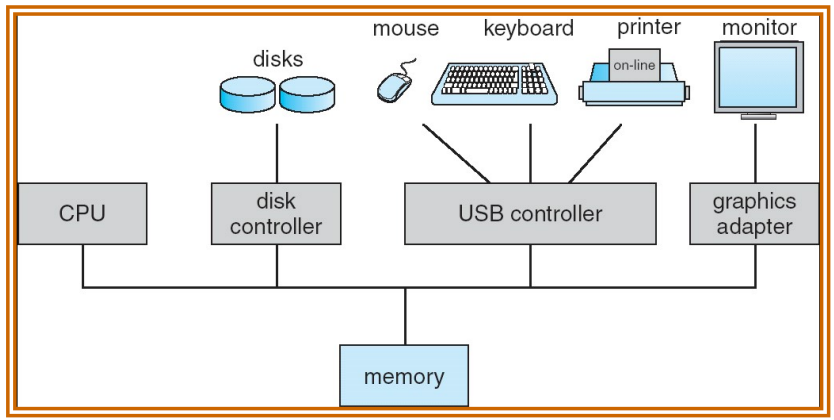
Moduli e Servizi

- Moduli e servizi tipici di un SO
  - Gestione dei processi
    - Un processo (unità attiva) è un programma (unità passiva) in esecuzione
    - Per essere eseguito richiede risorse
      - CPU, memoria, perferiche, etc.
    - Il SO deve
      - Creare, sospendere e cancellare un processo
      - Fornire meccanismi di comunicazione e sincronizzazione tra processi



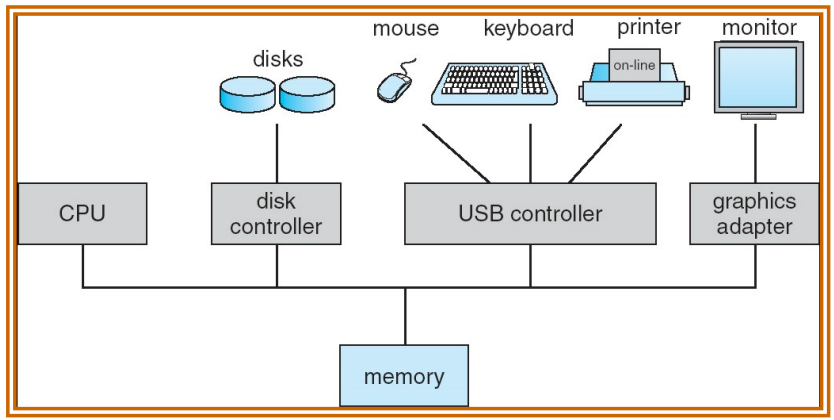
Moduli e Servizi

- Moduli e servizi tipici di un SO
  - Gestione della memoria principale
    - Dati e istruzioni sono immagazzinate nella memoria principale durante l'esecuzione di un programma
    - Logicamente la memoria è un vettore di elementi (word)
    - Il SO deve
      - Organizzare l'uso della memoria (quali word sono utilizzate e quali sono libere)
      - Decidere quali processi allocare/deallocare dalla memoria
      - Ottimizzare l'accesso ai dati da parte della CPU



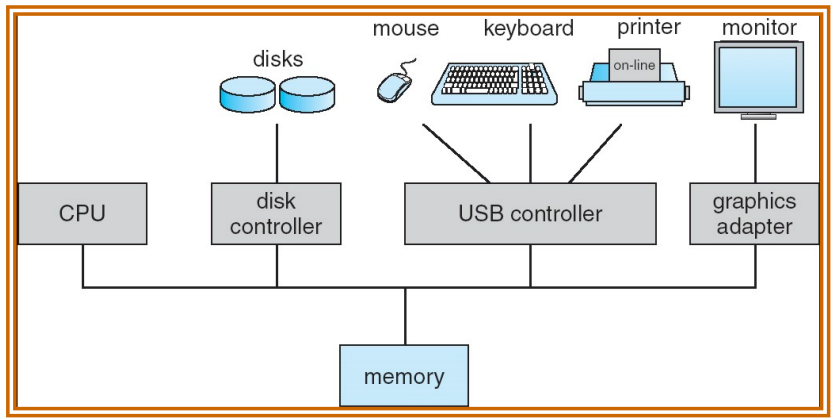
Moduli e Servizi

- Moduli e servizi tipici di un SO
  - Gestione della memoria secondaria
    - Dato che la memoria centrale è volatile e piccola i dati sono contenuti in maniera permanente su memoria di massa
    - Il SO deve
      - Organizzare le informazioni nello spazio disponibile
      - Allocare/deallocare lo spazio richiesto
      - Gestire lo spazio libero
      - Ottimizzare le scheduling delle operazioni di R/W



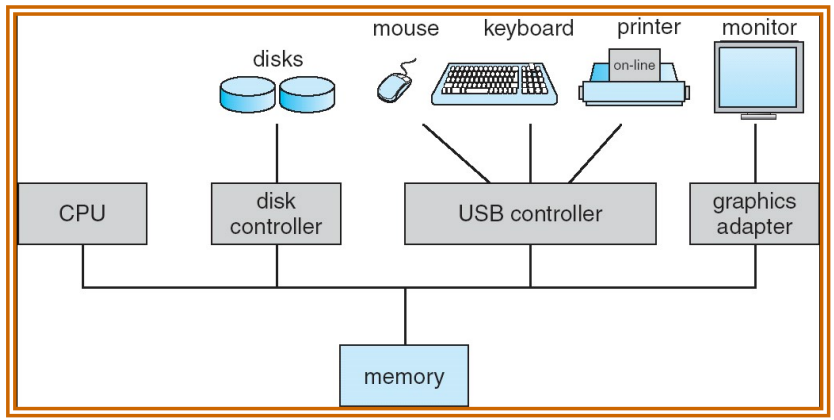
Moduli e Servizi

- Moduli e servizi tipici di un SO
  - Gestione dei dispositivi di I/O
    - I dispositivi di I/O non possono essere gestiti direttamente dagli utenti (complessità, driver, condivisione, etc.)
    - Il SO deve
      - Nascondere i dettagli di tali dispositivi agli utenti fornendo un'interfaccia generica tra ogni dispositivo e il relativo driver
      - Fornire operatività sui dispositivi



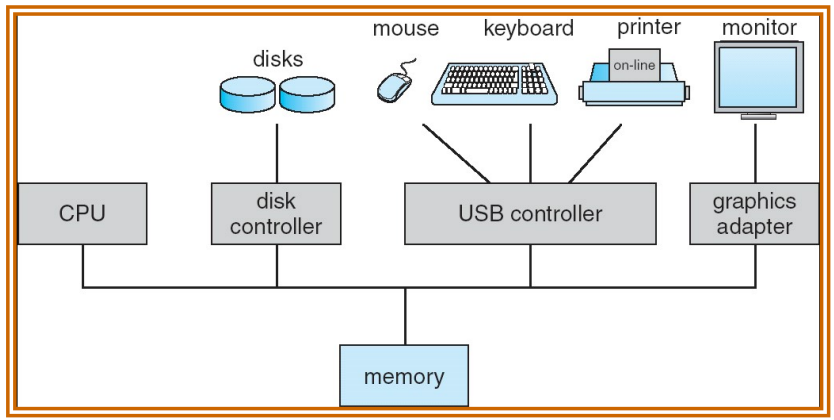
Moduli e Servizi

- Moduli e servizi tipici di un SO
  - Gestione file e filesystem
    - Le informazioni su memoria di massa sono organizzate in uno o più file-system, suddivisi in direttori, e contenenti file
    - Il SO deve
      - Creare, leggere, scrivere, cancellare direttori e file
      - Instaurare opportuni meccanismi di protezione di accesso, sicurezza da attacchi interni ed esterni
      - Ottimizzare le operazioni di R/W



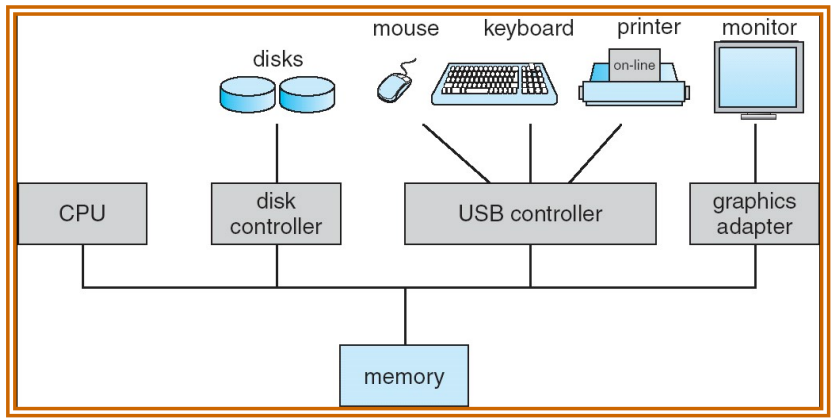
Moduli e Servizi

- Moduli e servizi tipici di un SO
  - Implementazione dei meccanismi di protezione
    - Con il termine **protezione** si indica il controllo sugli accessi da parte di utenti e processi alle risorse di sistema
    - Il SO deve
      - Definire i controlli che si vogliono instaurare
      - Distinguere tra utilizzo autorizzato e non autorizzato
      - Mantenere traccia di quali utenti usano le risorse del sistema



Moduli e Servizi

- Moduli e servizi tipici di un SO
  - Gestione delle reti e sistemi distribuiti
    - Una rete è un insieme di processori che non condividono memoria e clock e sono connessi da una rete
    - Il SO deve
      - Garantire l'accesso alle varie risorse del sistema
      - Incrementare le prestazioni del sistema di calcolo, controllare il numero di dispositivi disponibili, garantire l'affidabilità



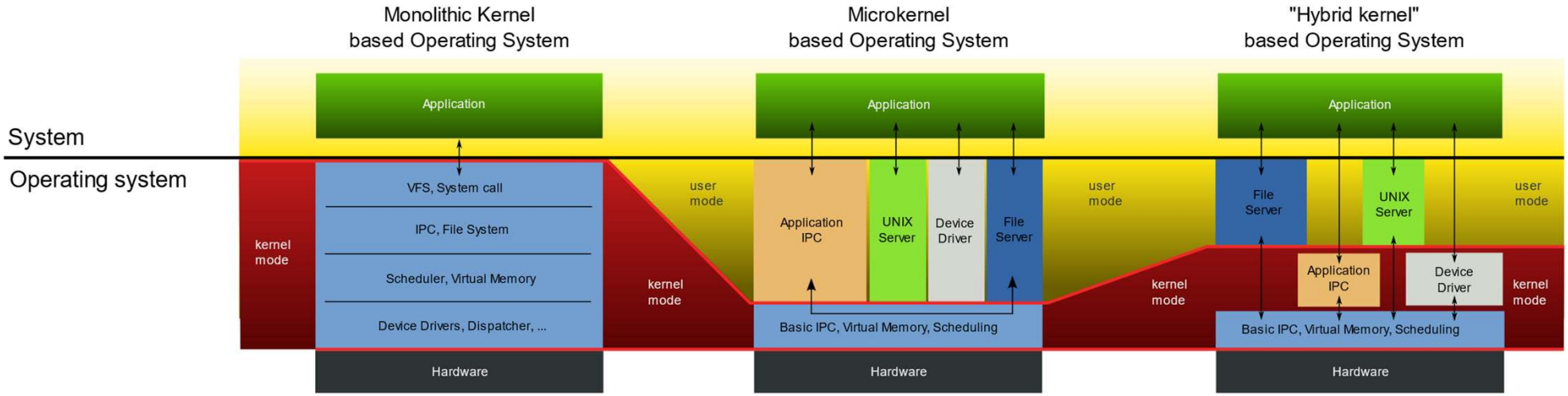
Terminologia e concetti base

- Terminologia e concetti base di un SO
  - Kernel, bootstrap, system-call
  - Login, shell
  - Filesystem, filename, pathname, home directory, root directory, working directory
  - Programma (sequenziale e concorrente), processo, thread, atomicità
  - Pipe
  - Deadlock, livelock, starvation, polling (busy waiting)

Terminologia e concetti base

- Kernel
  - Nucleo del sistema operativo
  - Funzionalità
    - Software che fornisce ai processi in esecuzione sull'elaboratore un accesso sicuro e controllato all'hardware
    - Ha la responsabilità di gestire i processori, il multi-tasking e la memoria
  - Unico programma in esecuzione per tutto il tempo
    - Tutti gli altri programmi sono programmi di sistema o applicativi

- Esistono diversi tipi di kernel
  - L'accesso diretto all'hardware può essere anche molto complesso
  - I kernel implementano uno o più tipi di astrazione dall'hardware



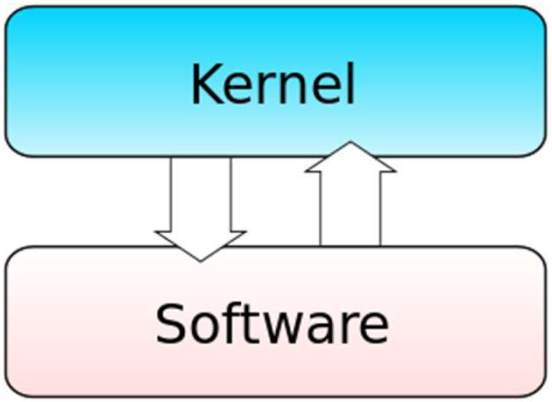
Terminologia e concetti base

Kernel monolitici

- I servizi sono realizzati tramite un insieme di chiamate di sistema realizzate da moduli separati ma la cui integrazione è molto stretta

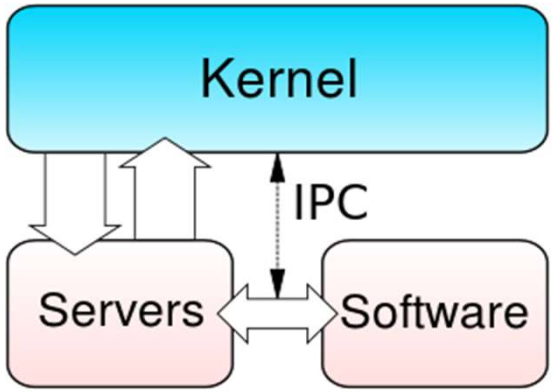
System call

- Svantaggi
  - Un problema su un modulo può bloccare l'intero sistema
  - Aggiungere un nuovo dispositivo hardware implica aggiungere il suo modulo al kernel e ricompilare l'intero kernel
- Vantaggi
  - La stretta integrazione interna dei componenti è estremamente efficiente
- Organizzazione comune
  - Unix, Linux, Open VMS, XTS-400



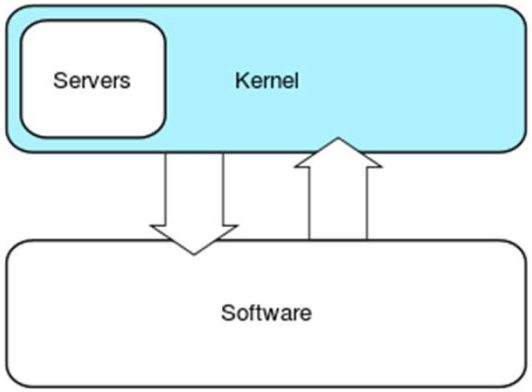
Terminologia e concetti base

- Microkernel
  - Definisce delle macchine virtuali molto semplici sopra l'hardware che implementano servizi minimali
  - Separa le implementazioni dei servizi di base dalle strutture operative del sistema operativi
  - Svantaggi
    - Lento a causa dell'elevato numero di chiamate di sistema e context switching
  - Vantaggi
    - Molto stabile
  - Organizzazione poco comune
    - March, L4, AmigaOS, Minix



Terminologia e concetti base

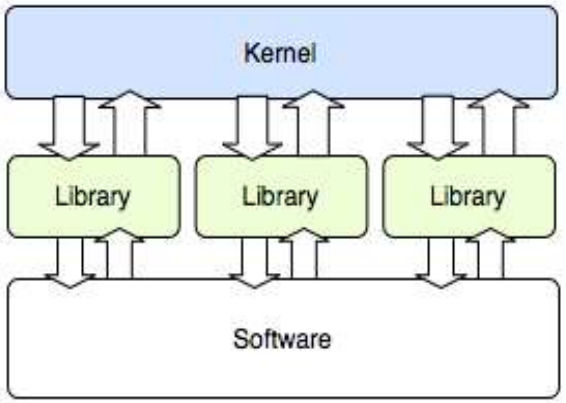
- Kernel ibridi
  - Approccio intermedio tra i due precedenti
    - Microkernel con codice aggiuntivo e "non essenziale" a livello di kernel che può essere eseguito molto rapidamente
  - Compromesso adottato da molti sviluppatori
  - Svantaggi
    - Prestazioni simili ai kernel monolitici
  - Vantaggi
    - Integra vantaggi dei monolitici e dei microkernel
  - Organizzazione comune
    - Windows NT, Netware, XNU Kernel di Mac OS X, Dragonfly BSD



Terminologia e concetti base

Esokernel

- Noti come "sistemi operativi verticali"
  - Approccio radicalmente differente alla progettazione dei sistemi operativi
  - Separano la "protezione dalla gestione"
- Estremamente piccoli e compatti, in quanto le loro funzionalità sono arbitrariamente limitate alla protezione e al multiplexing delle risorse
- Vantaggi
  - Minore astrazione dell'hardware
- Svantaggi
  - Implicano maggior lavoro nello sviluppo di applicazioni
- Esempi
  - Nemesis, ExOS



Terminologia e concetti base

- Bootstrap (bootstrap o booting program)
  - Programma di inizializzazione
  - Carica il kernel in memoria centrale all'accensione del computer
  - Esegue tale kernel, permettendo una inizializzazione corretta di tutti gli aspetti principali
  - Il programma di bootstrap è usualmente
    - Memorizzato in ROM o EEPROM (firmware)
    - Caricato al power-up o al reboot

Terminologia e concetti base

- System call
  - Forniscono l'interfaccia ai servizi forniti dal sistema operativo stesso ovvero sono gli entry-point del SO
    - Spesso sono implementate in assembler
    - Spesso vi si accede attraverso un Application Program Interface (API) di alto livello
      - Win32/64 API (per Windows)
      - POSIX API (per UNIX, Linux, MAC OS X)
      - JAVA API (per la Java Virtual Machine)
  - Quante system call esistono in un SO?
    - UNIX 4.4BSD: circa 110
    - Linux: tra 240 e 260
    - UNIX FreeBSD: circa 320

Terminologia e concetti base

- La differenza tra una system call e una funzione di libreria è sottile
  - Entrambe forniscono servizi all'utente
  - Per ogni system call esistono normalmente una o più funzioni di alto livello (e.g., C) con lo stesso nome
  - Le funzioni possono essere sostituite o modificate le system call no
  - Le system call forniscono usualmente funzionalità di base, mentre le funzioni di libreria risultano maggiormente elaborate e hanno un numero maggiore di parametri

Esempi di System Call

- Le system call assumo aspetti diversi
  - Stile API POSIX

Identica su MAC OS X (read, pread, readv)

UNIX

```
int read (int fd, void *buffer, size_t nbytes);
```

  - Stile API Win32/64

Windows

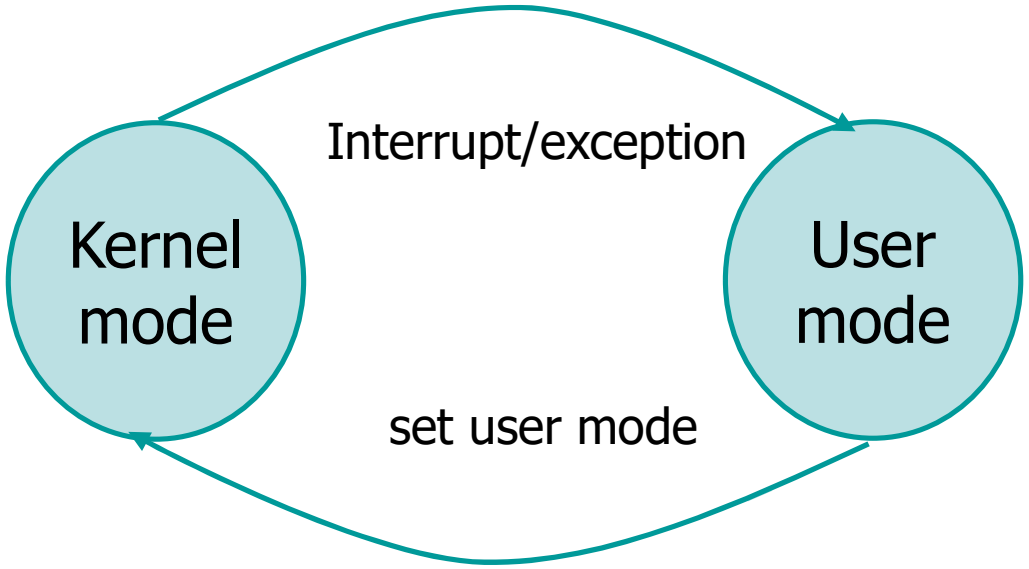
```
BOOL ReadFile (
   HANDLE fileHandle,
   LPVOID dataBuffer,
   DWORD numberOfByteToRead,
   LPDWORD numberOfByteRead,
   LPOVERLAPPED overlappedDataStructure
);
```

Terminologia e concetti base

- Il SO si protegge lavorando in **dual-mode**
  - Gli utenti lavorano in modalità Utente
    - Bit di modo = 1
  - Il kernel lavora in modalità protetta
    - Bit di modo = 0

Nella PWS (Program Status Word)

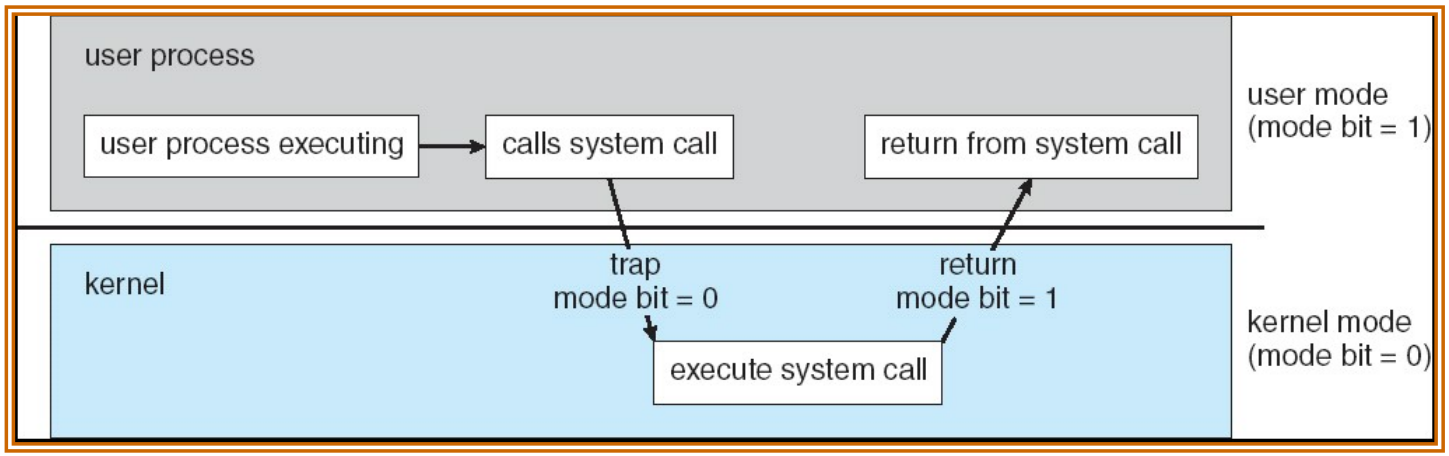
Le istruzioni privilegiate possono essere eseguite solo in kernel mode



Il dual-mode assicura che lo user non possa assumere il controllo del computer in kernel mode

Terminologia e concetti base

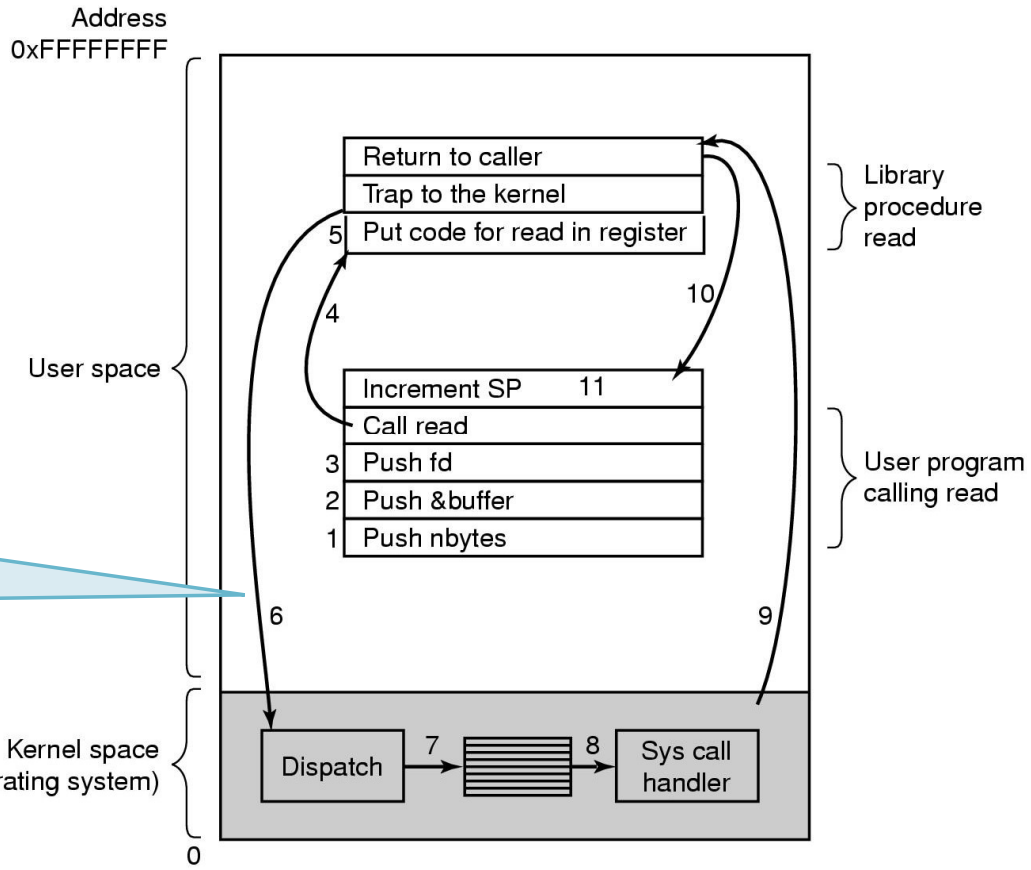
- In SO passa al kernel mode all'arrivo di un interrupt, una eccezione, un fault, etc.
- Come si esegue una operazione privilegiata (e.g., tutte le operazione di I/O sono privilegiate) ?
  - Le system call sono usualmente effettuate attraverso una interruzione software o **trap** (che attiva il codice della system call richiamata)
  - Si passa così in kernel mode



Terminologia e concetti base

- L'esecuzione di una system call implica l'esecuzione di 11 passi

```
read (fd, buffer, nbytes);
```



Trap:
Passaggio in modalità kernel
Utilizzo di un indice per tabella indirizzi

Terminologia e concetti base

- Esempi di funzioni di libreria e system call
  - La funzione **printf** utilizza la system call **write**
  - La funzione di allocazione **malloc** plausibilmente richiama la system call **sbrk**
  - Le funzioni che gestiscono la data e l'ora fanno riferimento a un'unica system call di nome **time**
    - **time** fornisce il numero di secondi trascorsi dal 01.01.1970
    - Le funzioni che la richiamano forniscono data e ora con formati diversi

Terminologia e concetti base

- Elenco delle più comuni system call UNIX/Linux
  - Gestione processi
    - fork, wait, exec, exit, kill
  - Gestione file
    - open, close, read, write, lseek, stat
  - Gestione direttori
    - mkdir, rmdir, link, unlink, mount, umount, chdir, chmod

Terminologia e concetti base

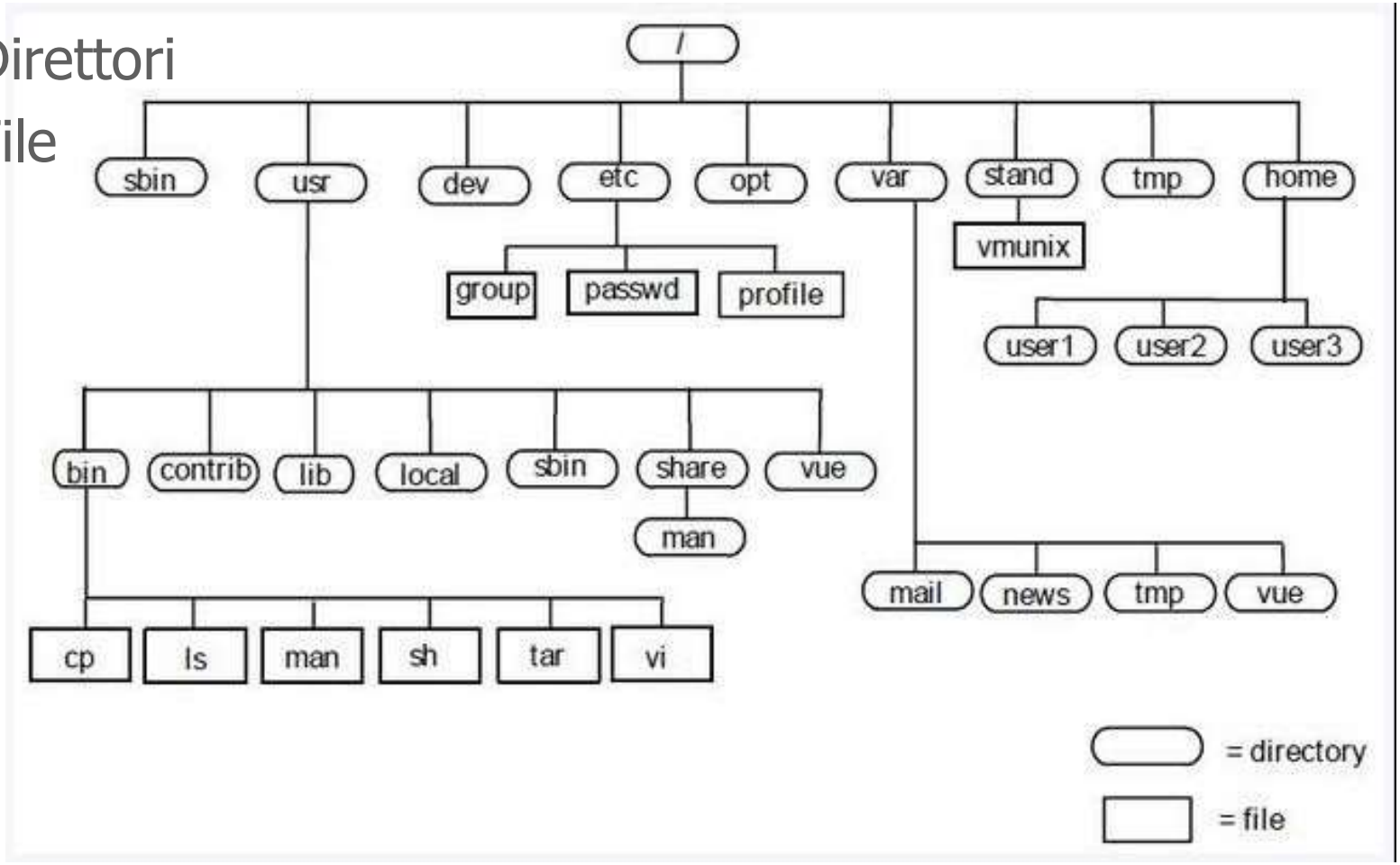
- Login
  - Il login (spesso detto procedura di **autenticazione**) è il termine utilizzato per indicare la procedura di accesso a un sistema informative o a una sua applicazione
  - Per effettuare un login occorre usualmente fornire
    - Username
    - Password
      - Le password sono usualmente codificate nel file /etc/passwd

Terminologia e concetti base

- Shell
  - Interfacccia UNIX per eccellenza, i.e., interprete "command line"
  - Non fa parte del SO
  - Legge i comandi utente e li esegue
    - I comandi possono essere digitati su terminale
    - Letti da un file eseguibile (interpretati) detti file di "script"
  - Esistono diverse shell di uso comune
    - Bourne shell (sh)
    - Bourne again shell (bash)
    - Etc.

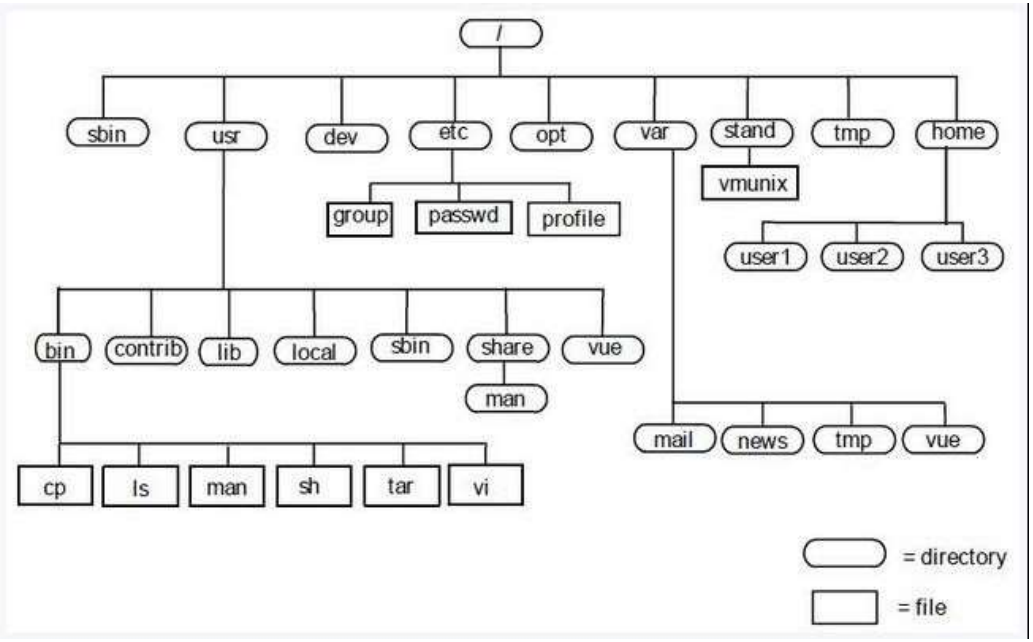
Terminologia e concetti base

- Filesystem
  - Struttura gerarchica a **grafo aciclico** in cui sono organizzati
    - Direttori
    - File



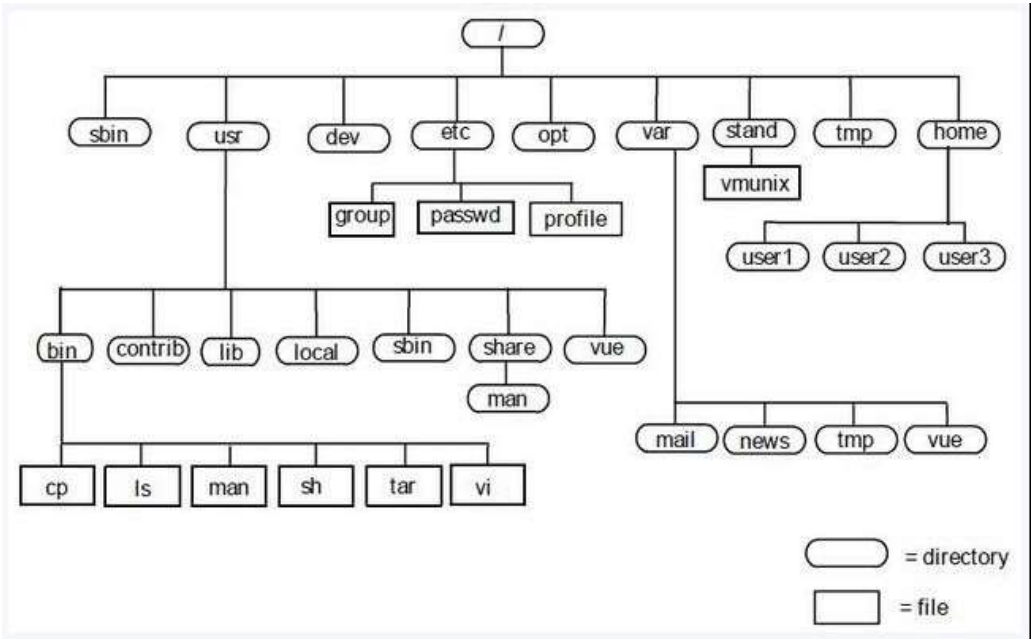
Terminologia e concetti base

- Filename
  - Esistono regole di composizione
  - In UNIX gli unici caratteri che non possono essere inseriti in un nome di file sono
    - Lo slash "/"
    - Il carattere "null"



Terminologia e concetti base

- Pathname
  - Una sequenza di nomi separati da slash "/"
    - Per esempio, /run/user/, /home/quer/, etc.
  - I path-name possono essere specificati in maniera
    - Assoluta (vedere Root Directory)
    - Relativa (vedere Working Directory)
  - I carattere/caratteri
    - "." indica il direttorio corrente
    - ".." indicano il direttorio padre



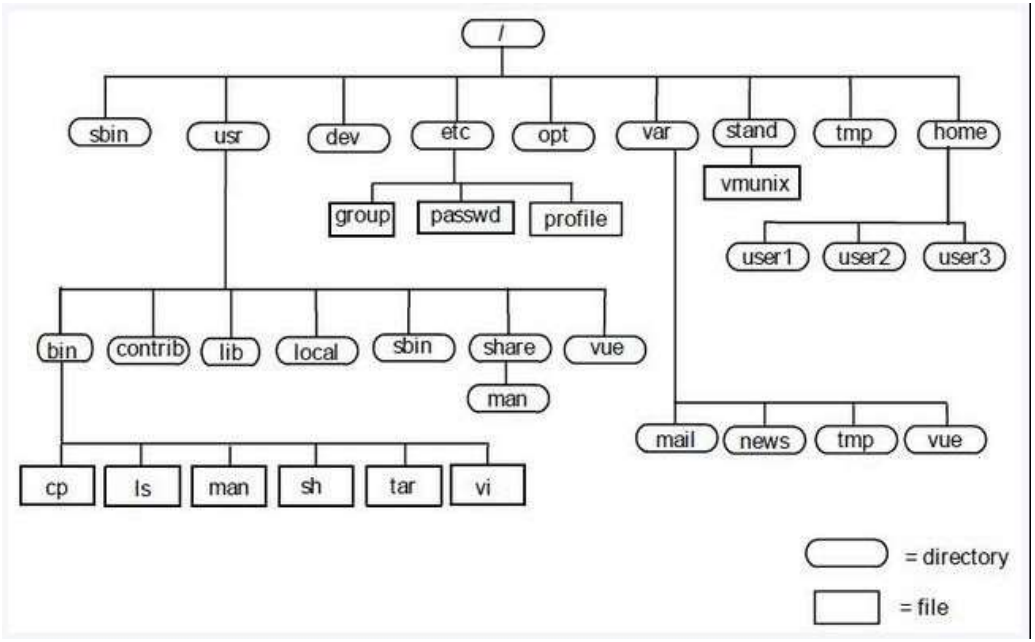
Terminologia e concetti base

- Home directory
  - Direttorio a cui si accede una volta fatto il **login**
  - Contiene il materiale dello **user** che ha fatto il login
  - Individuata con la tilde ~ nei sistemi UNIX-like
    - Per esempio, per lo user foo la home directory si trova usualmente

      in /home/foo

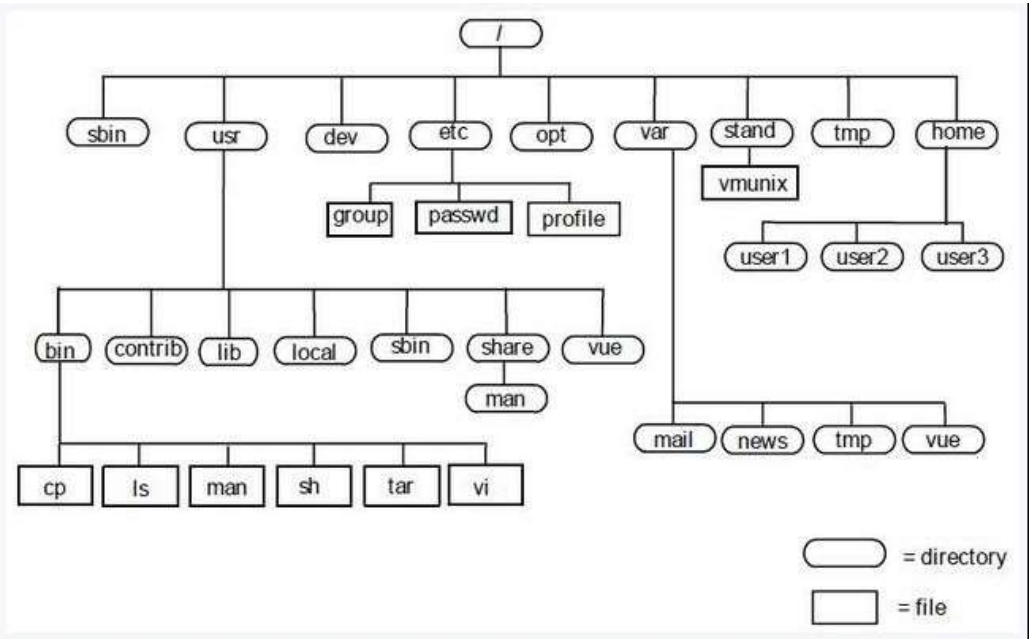
      e corrisponde a

      ~ per tale user



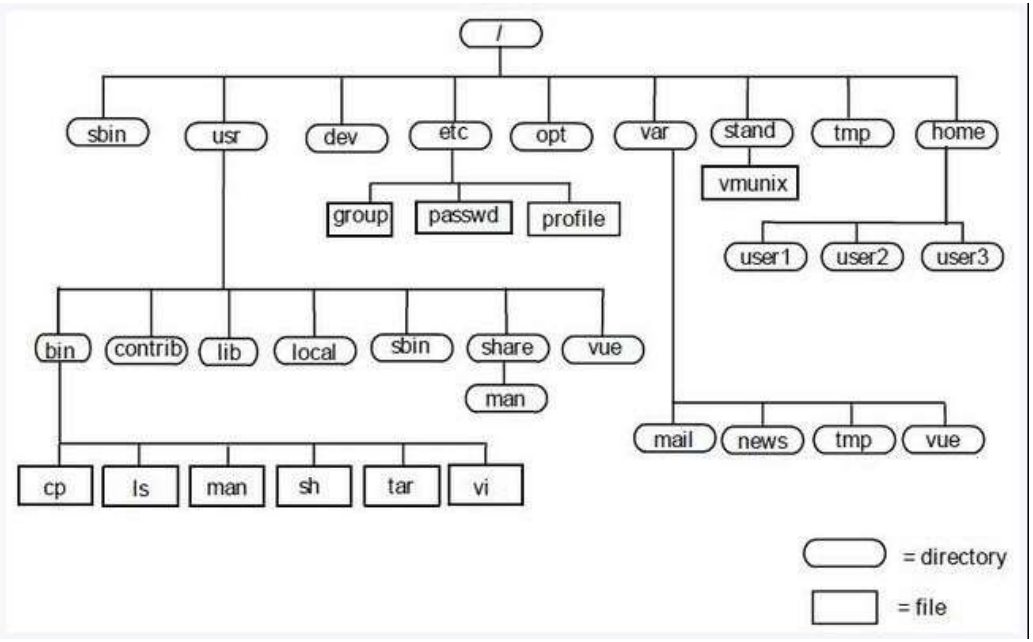
Terminologia e concetti base

- Root directory
  - Direttorio principale
  - Radice dell'albero direttori
  - Punto di origine per interpretare i path assoluti



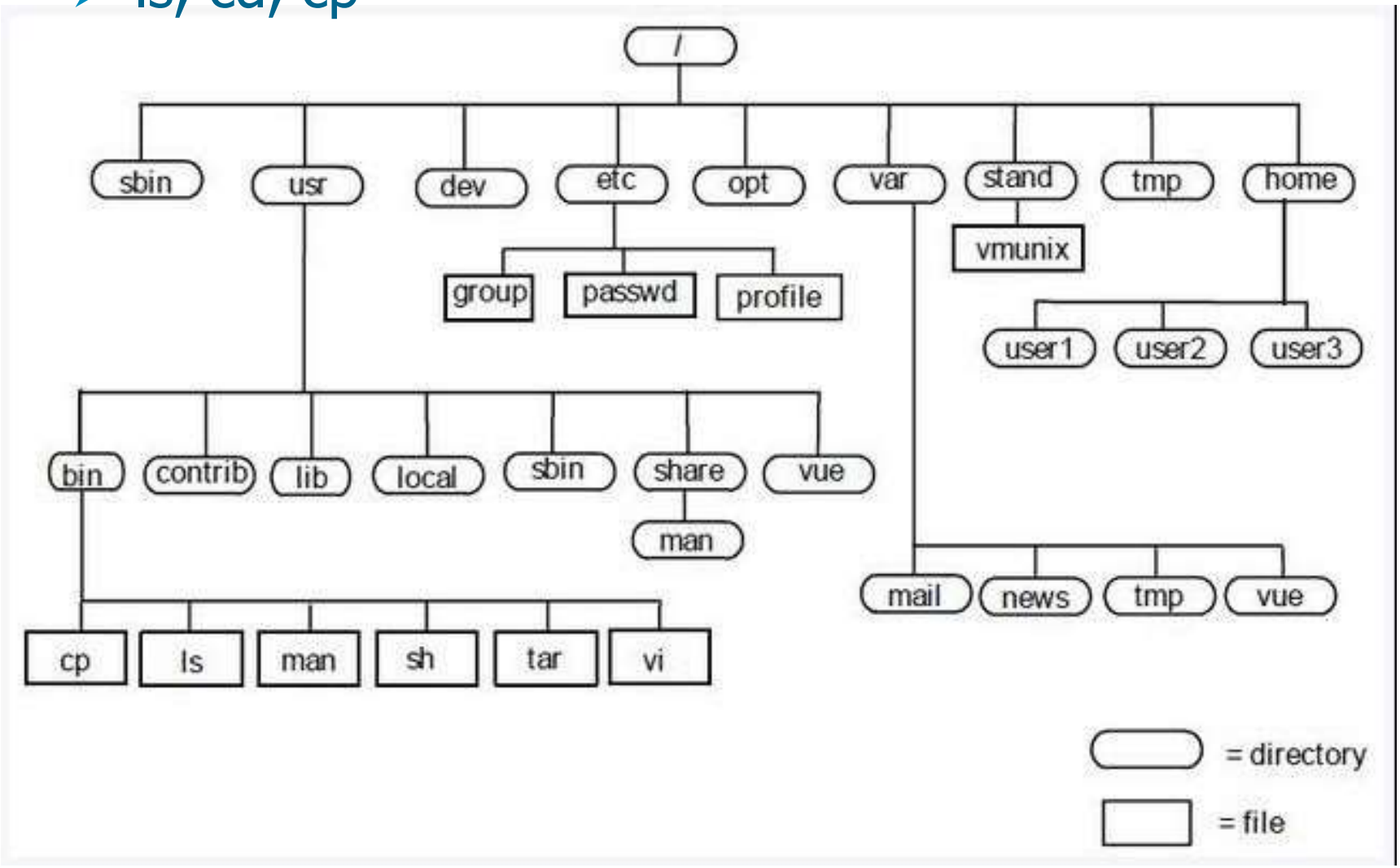
Terminologia e concetti base

- Working directory
  - Punto di origine per interpretare i path relativi
  - Inizialmente pari alla home directory
  - Può essere modificata seguendo la struttura del file-system
  - Posseduta da ogni processo
  - Ci si riferisce automaticamente qualora non si specifichi un path



Esempio

- Esempio di accesso tramite i comandi
  - ls, cd, cp

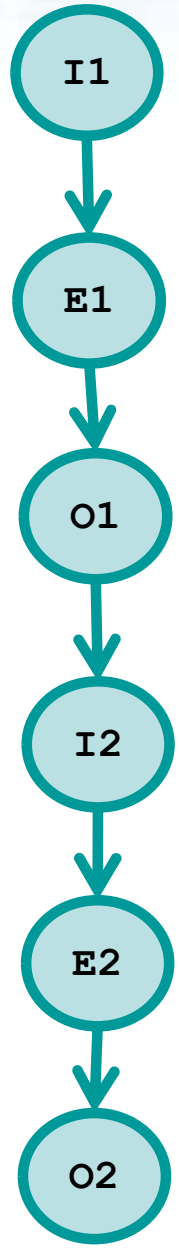


Terminologia e concetti base

- Programma
  - Entità passiva
  - File eseguibile che risiede su disco
  - Specifica un insieme di **operazioni** (più o meno elementari) per realizzare un definite procedimento (algoritmo)

Terminologia e concetti base

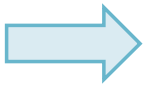
- Programma sequenziale
  - Indica operazioni da eseguire in sequenza
    - Esempio: Input, Elaborazioni, Output
  - Ogni nuova istruzione inizia al termine della precedente
    - Secondo il ciclo di fetch, decode, e execute



Terminologia e concetti base

- Una operazione si dice **atomica** se nessun processore può interromperla
  - Esempio

```
x++;
```

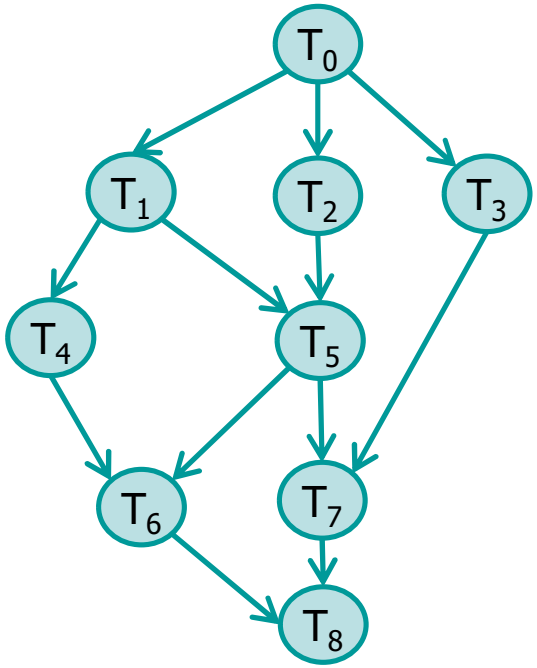


```
add BYPE PTR [0x20], 1
```

    - Per aggiungere 1 a una locazione di memoria è ragionevole fare una copia del valore in un registro
    - Se, nello stesso intervallo di tempo, un altro processo eseguie la stessa operazione, uno dei due incrementi può andare perduto
    - L'atomicità garantisce questo non possa avvenire
    - Perché non garantire sempre l'atomicità?
      - L'atomicità è costosa da garantire e rallenterebbe l'intero computer

Terminologia e concetti base

- Programma concorrente o parallelo
  - Individua operazioni che possono procedere in parallelo
  - Una operazione può essere eseguita senza attendere il completamento della precedente
  - La relazione temporale delle operazioni può essere specificata tramite un **grafo di precedenza**



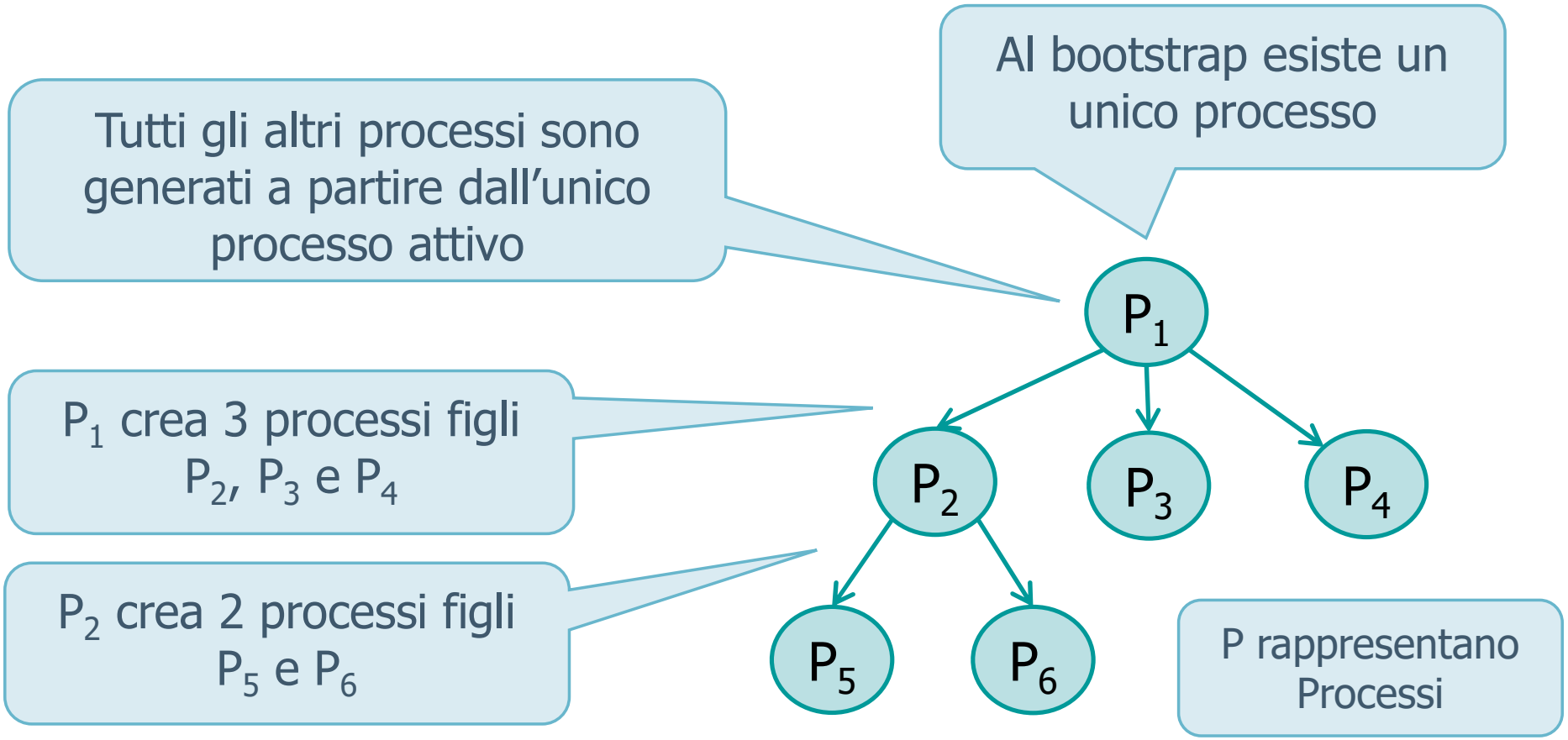
T rappresentano Task
(processi, thread, blocchi di istruzioni, singole istruzioni, etc.)

Terminologia e concetti base

- Processo
  - Un programma in esecuzione (program counter, registri, variabili, etc.)
  - Entità attiva
  - Nei sistemi UNIX ogni processo è caratterizzato da un identificatore intero univoco (non negativo)

Terminologia e concetti base

- La relazione tra i processi è rappresentata mediante un **albero dei processi**



Terminologia e concetti base

- Thread
  - Un processo "raggruppa" le risorse
  - Un thread individual un flusso di esecuzione
    - Un processo può avere al suo interno uno o più flussi di controllo in esecuzione
    - Ciascun flussi di esecuzione è un thread
      - Ogni thread corrisponde a una entità schedulata separatamente
  - Ogni thread viene individuato tramite un identificatore (locale al processo)

Thread o processo leggero

Terminologia e concetti base

- Pipe
  - Sono la più vecchia forma di comunicazione tra processi diversi messa a disposizione dai sistemi UNIX
  - Una pipe è un **flusso dati** tra due processi

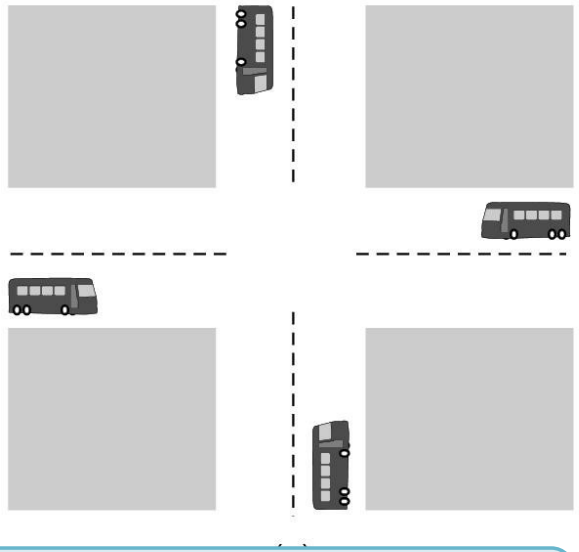
  - Nella loro eccezione più semplice è un canale di comunicazione **half-duplex**
    - Comunicazione da P_1 a P_2 oppure da P_2 a P_1

Comunicazione **simplex**: effettuabile solo da P_1 a P_2

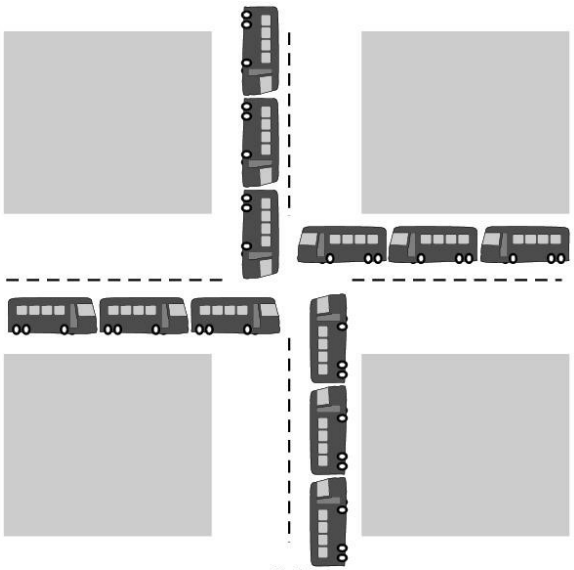
Comunicazione **full-duplex**: effettuabile da P_1 a P_2 e viceversa contemporaneamente

Terminologia e concetti base

- Deadlock (stallo, impasse)
  - Un insieme di entità attendono il verificarsi di un evento che può essere causato solo da un'altra entità dell'insieme
  - Esempi



Stallo potenziale



Stallo conclamato

Terminologia e concetti base

- Livelock (stallo attivo)
  - Situazione simile al deadlock in cui i le entità non sono effettivamente bloccate ma di fatto non fanno alcun progresso
  - Esempi
    - Due persone si incontrano in un corridoio e cercando di passare si spostano ripetutamente da un lato all'altro del corridoio stesso; non procedono lungo il corridoio ma non sono bloccate
    - Due unità effettuano del **polling** (**busy waiting**) per verificare lo stato dell'altro e non fanno progressi (livelock mutuo) ma non sono in deadlock visto che comunque effettuano il polling

Terminologia e concetti base

- Starvation (fame, inedia)
  - A una entità viene ripetutamente rifiutato l'accesso a una risorsa necessaria al suo progresso
  - Esempi
    - Starvation **non** implica deadlock
      - Mentre una entità è in condizione di starvation le altre possono progredire
    - Deadlock **implica** starvation
      - Nessuna entità procede quindi tutte sono in starvation