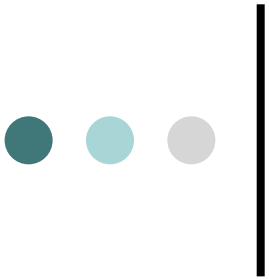
Modulo di Sistemi Operativi per il corso di Master
RISS: Ricerca e Innovazione nelle Scienze della Salute
Unisa, 17-26 Luglio 2012



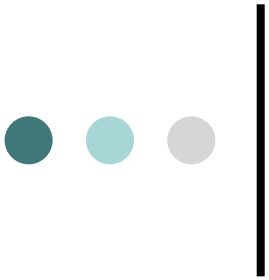
Sistemi Operativi

Funzioni e strategie di progettazione: dai kernel monolitici alle macchine virtuali



Servizi e strutture

- Descrizione dei servizi offerti all'utente dal SO
  - Utente generico
  - Programmatore
- Esame delle possibili strutture di un SO
  - Monolitica
  - Stratificata
  - A microkernel
  - A moduli caricabili dinamicamente
  - Macchine virtuali
- Installazione ed adattamento. Procedura d'avvio



Servizi per l'utente generico

Servizi dei sistemi operativi

- Il SO fornisce funzioni utili per l'utente quali:
  - **Esecuzione dei programmi** – Il SO carica in memoria ed esegue i programmi e ne termina l'esecuzione in modo normale o anormale, indicandone l'errore
  - **Operazioni di I/O** - Un programma in esecuzione può richiedere I/O, file o dispositivi di I/O
  - **Gestione del file-system** - I programmi necessitano di leggere e scrivere file e directory, crearne e cancellarne, accedere alle informazioni di gestione del file, gestire i permessi

Servizi dei sistemi operativi

- **Comunicazioni** – I processi possono scambiarsi informazioni, sullo stesso computer o tra computer su una rete

- **Rilevazione dell'errore** – il SO deve essere costantemente informato sugli errori
  - CPU, hardware della memoria, I/O device, programmi utente
  - Per ogni tipo di errore, il SO dovrebbe prendere le azioni appropriate per assicurare computazioni corrette e consistenti
  - Strumenti di supporto per il debug incrementano le possibilità dell'utente e dei programmatori di usare efficientemente il SO

Servizi impliciti

- **Allocazione risorse** – Quando più utenti o processi vengono eseguiti concorrentemente, le risorse debbono essere allocate ad ognuno di essi
  - Risorse - cicli CPU, memoria, file, dispositivi I/O
- **Contabilizzazione** – Tenere traccia di quali utenti (e quanto) usano certe risorse
- **Protezione e sicurezza** – I proprietari di dati memorizzati in un sistema multiutente o connesso alla rete desiderano controllare l'uso dei propri dati ed esser sicuri che i processi non interferiscano
  - **Protezione**: tutti gli accessi alle risorse del sistema sono controllati
  - **Sicurezza**: autenticazione degli utenti contro attacchi esterni
  - "una catena è tanto forte quanto il suo anello più debole".

Interfaccia utente CLI

Le interfacce a linea di comando CLI permettono l'invio diretto di comandi

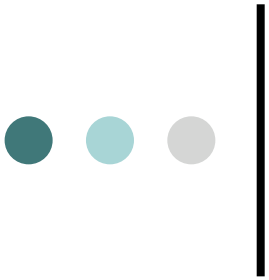
- Caratteristiche multiple e opzioni – **shell**
  - A volte i comandi sono built-in (implementati nel kernel), altre sono semplici nomi di programmi
  - Con il secondo approccio, l'aggiunta di nuove caratteristiche non richiede la modifica della shell

Unix: Bourne Shell, C Shell, Bourne-again Shell, Korn Shell

Interfaccia utente GUI

- Le interfacce grafiche forniscono **desktop** amichevoli
  - I comandi sono forniti tramite mouse, tastiera, monitor
  - **Icone** rappresentano file, programmi, azioni …
  - La pressione dei tasti del mouse sugli oggetti dell'interfaccia causa varie azioni (recupero informazioni, opzioni, esecuzioni di funzioni, apertura di directory)
  - Inventate a Xerox PARC, comuni con Apple Mac

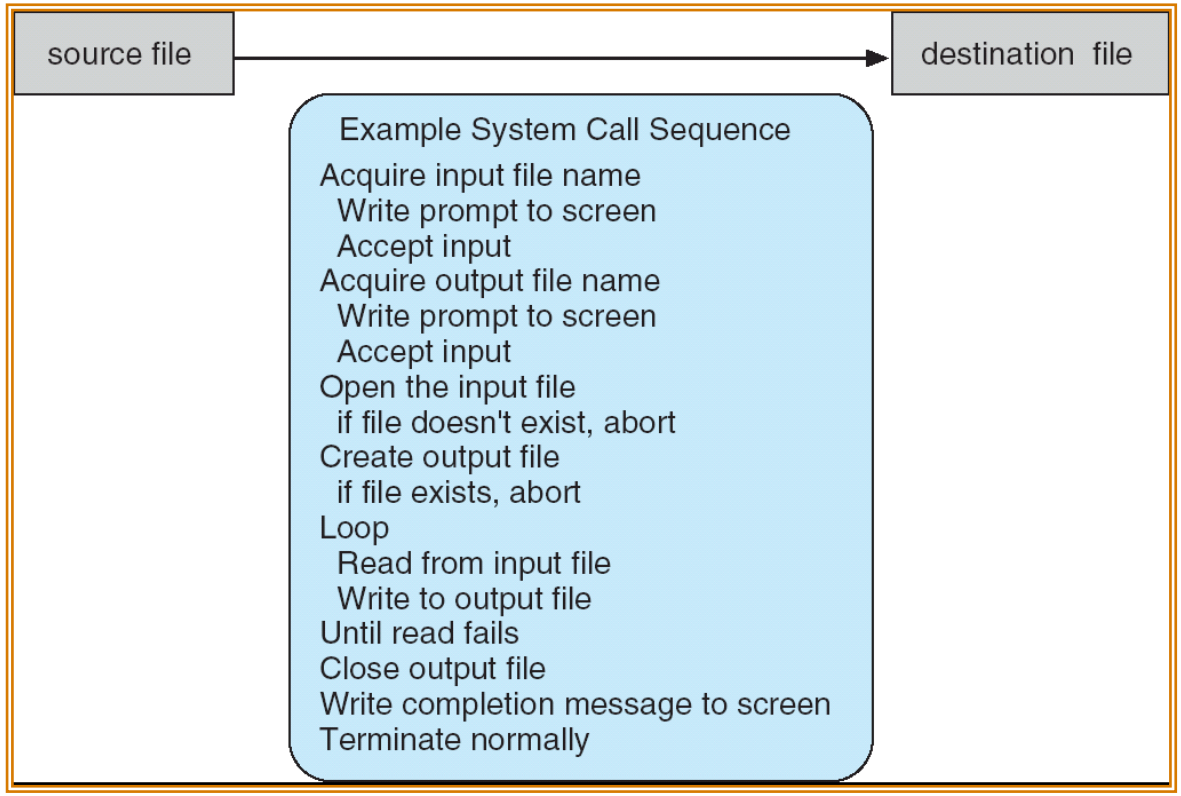
- Molti sistemi oggi includono interfacce sia CLI che GUI
  - Microsoft Windows offre una GUI ed una CLI
  - Apple Mac OS X offre una GUI (Aqua), che poggia su un kernel UNIX, e mette a disposizione le shell UNIX
  - Solaris offre una CLI e opzionalmente GUI (Java Desktop, KDE)



Servizi per il programmatore

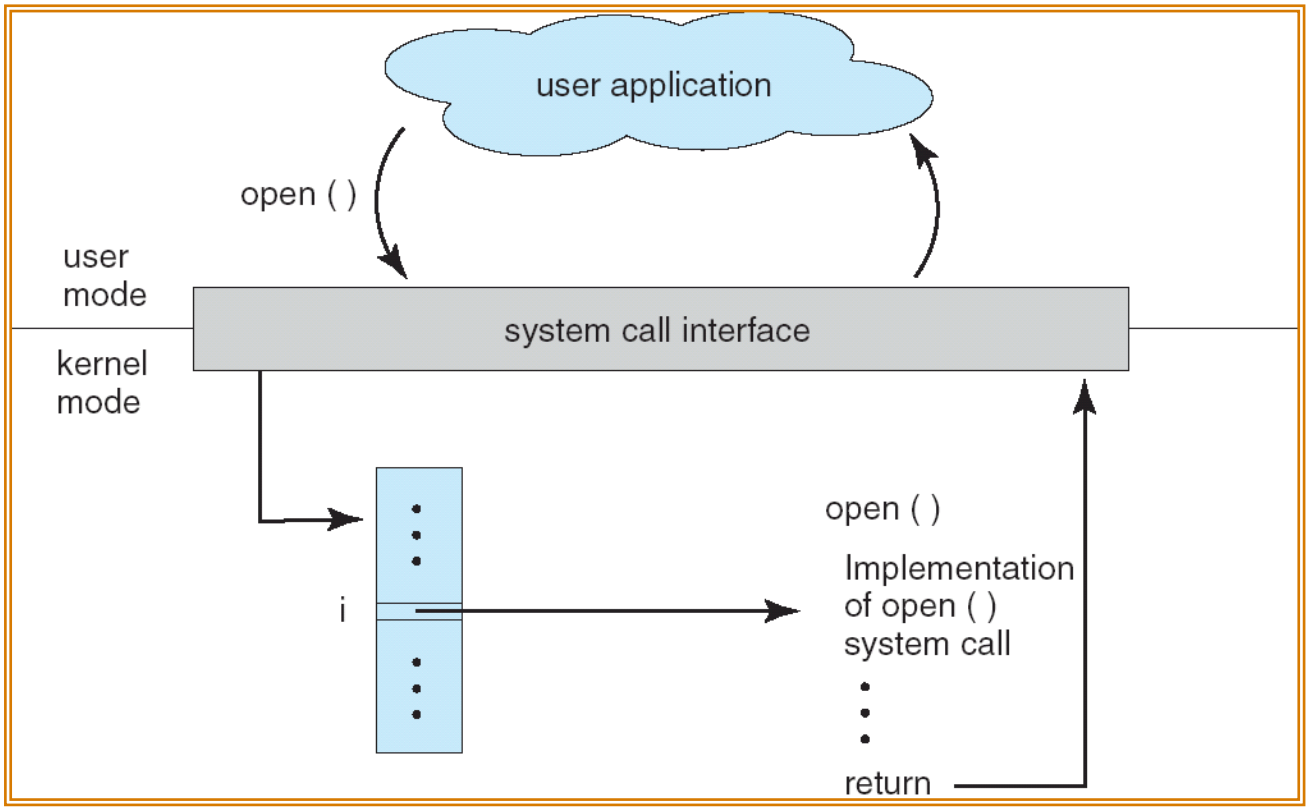
Chiamate di sistema – System Call

- Interfaccia per i programmatori ai servizi offerti dal SO



Gestione di chiamate di sistema

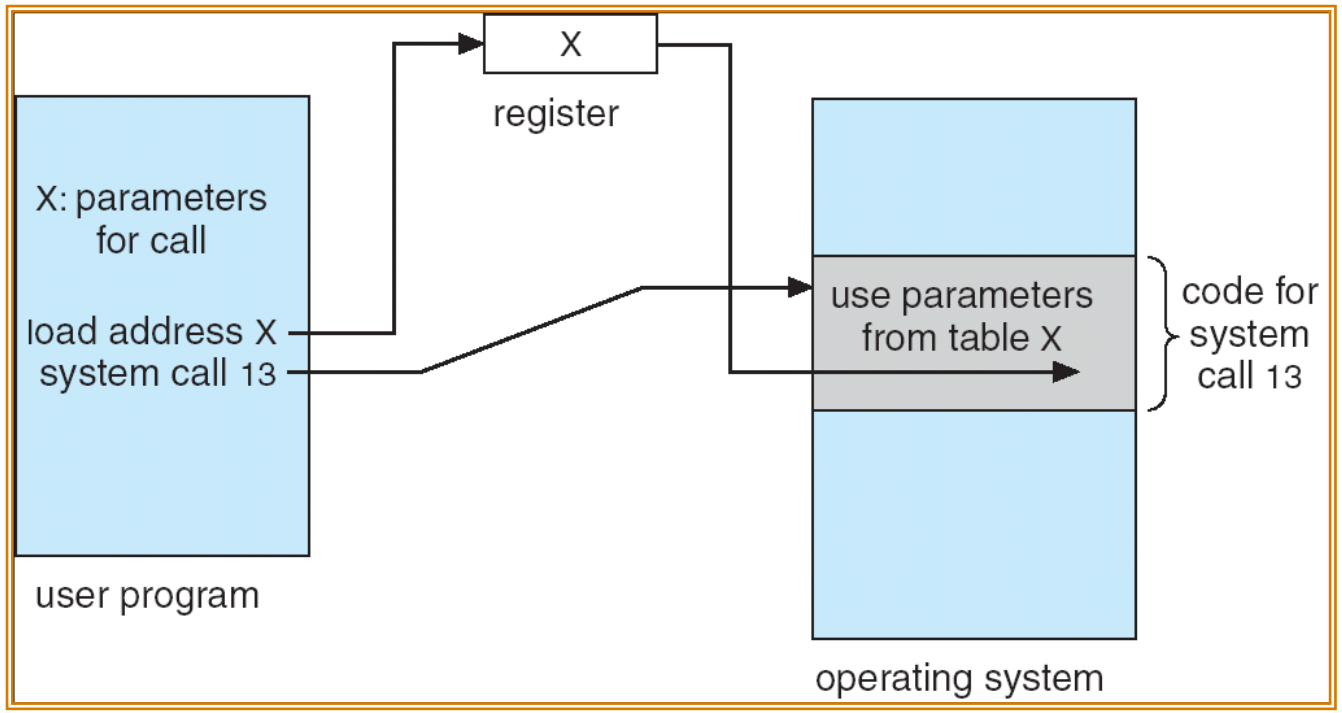
- Tipicamente ad ogni chiamata di sistema è associato un numero
  - Il sistema mantiene una tabella indicizzata da questi numeri



Passaggio dei parametri

- Spesso sono richieste informazioni aggiuntive all'identificativo della system call
  - Il tipo e la quantità di informazione dipendono dal SO e dalla call
- Tre metodi sono usati per passare i parametri al SO
  - Passaggio dei parametri nei **registri**
  - Parametri memorizzati in un **blocco**, o **tabella in memoria**, e l'indirizzo del blocco viene passato al SO in un registro
  - I parametri vengono posizionati nello **stack** dal programma e prelevati dal SO

Passaggio di parametri attraverso una tabella



API e Chiamate di sistema

- I programmi vi accedono principalmente tramite **Application Programming Interface (API)** piuttosto che chiamate dirette

- Tre API comuni sono la **Win32 API** per Windows, la **POSIX API** per sistemi POSIX-based (che virtualmente includono tutte le versioni di UNIX, Linux, e Mac OS X), e la **Java API** per la Java virtual machine (JVM)

- Perchè usare API invece di system call direttamente? Portabilità dei programmi, dettagli delle system call reali …

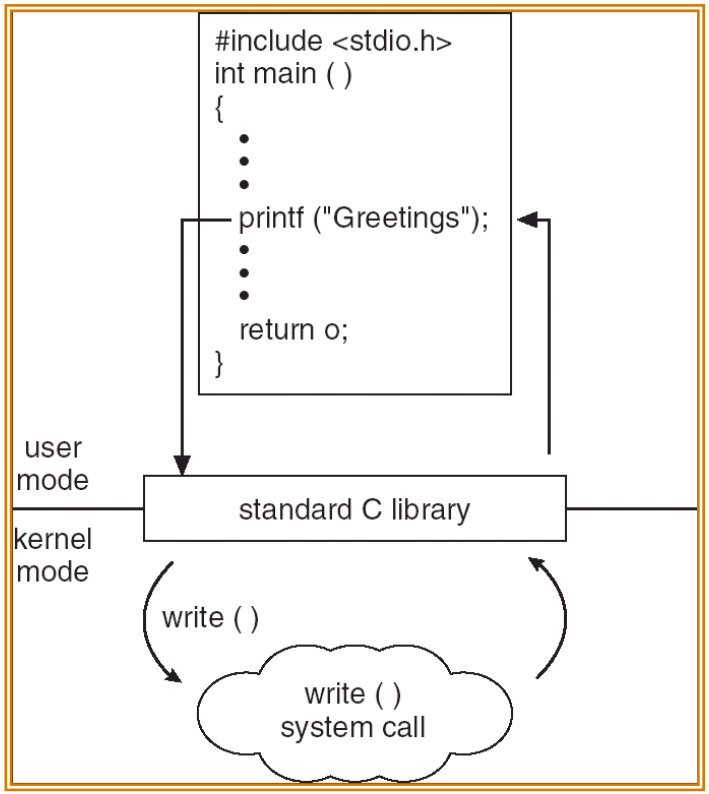
API – System Call – Relazione con il SO

- L'interfaccia (API) alle chiamate di sistema invoca l'opportuna chiamata di sistema e restituisce lo stato e qualunque valore di ritorno della chiamata stessa

- Il chiamante deve
  - semplicemente seguire le specifiche dell'API e capire cosa il SO fa a seguito di una chiamata
  - l'API nasconde al programmatore molti dettagli implementativi, gestiti dal sistema di supporto run-time (insieme di funzioni della libreria inclusa con il compilatore)

Esempio dalla Libreria Standard C

- Programma C che usa la funzione di libreria printf(), la quale invoca la chiamata di sistema write()



Categorie di system call

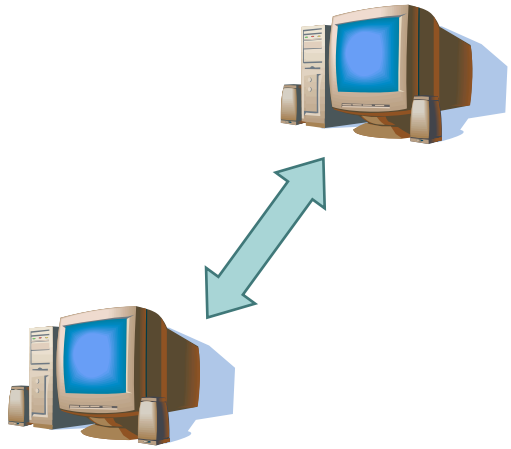
- Controllo dei processi
- Gestione dei file
- Gestione dei dispositivi
- Gestione delle informazioni di stato
- Comunicazioni

Categorie di system call

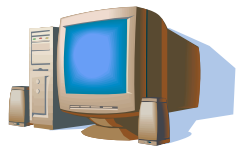
- **Processi**
  - end, abort
  - load, execute
  - create process, terminate process
  - get process attribute, set process attributes
  - wait for time
  - wait event, signal event
  - allocate memory, free memory
- **File**
  - create file, delete file
  - open, close
  - read, write, reposition
  - get file attribute, set file attributes
- **Informazioni**
  - get time, get date, set time, set date
  - get system data, set system data
  - get process, file, or device attributes
  - set process, file, or device attributes
- **Dispositivi I/O**
  - request device, release device
  - read, write, reposition
  - get device attribute, set device attributes
  - attach device, detach device

Categorie di system call

- **Scambio di messaggi**
  - gethostid – IP address
  - getprocessid - PID
  - open connection
  - accept connection
  - send/receive
  - close connection



- **Memoria condivisa**
  - shared memory create
  - shared memory attach

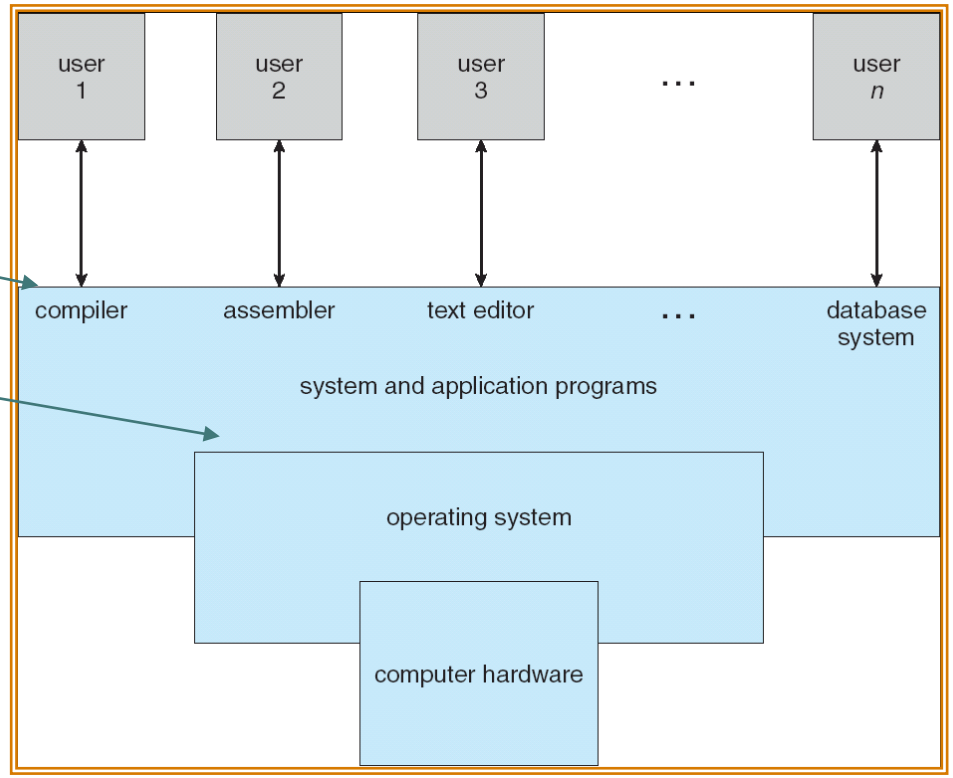


Programmi di Sistema

- Alcuni sono semplici interfacce alle chiamate di sistema. Altri sono considerevolmente più complessi.

Programmi di sistema

System call



Programmi di sistema

- **Gestione dei File** - Create, delete, copy, rename, print, dump, list, e programmi per manipolare file e directory

- **Gestione informazioni di stato**
  - Alcuni chiedono al SO informazioni quali data, ora, quantità di memoria disponibile, spazio su disco, numero di utenti
  - Altri forniscono dettagli sulle prestazioni, informazioni sulle operazioni di log e il debug
  - Tipicamente, questi programmi formattano e stampano le informazioni al terminale o su altri dispositivi di output
  - Alcuni sistemi implementano un file registro, usato per memorizzare e recuperare informazioni di configurazione del SO

Programmi di Sistema

- **Editing dei File**
  - Editor di testo per creare e modificare file
  - Comandi speciali per cercare contenuti di file o elaborare il testo
- **Supporto ai linguaggi di programmazione** - Compilatori, assemblatori, debugger e interpreti
- **Caricamento ed esecuzione dei programmi** – Caricatori assoluti, caricatori rilocabili, linker, e caricatori di overlay, sistemi di debug per linguaggi ad alto livello
- **Comunicazioni** – Forniscono un meccanismo per creare connessioni virtuali tra processi, utenti, e sistemi di calcolatori
  - Permettono agli utenti di inviare messaggi ad un altro schermo, sfogliare pagine web, inviare messaggi di posta elettronica, login remoto, trasferire file da una macchina ad un'altra

Solaris 10 dtrace

```
# ./all.d 'pgrep xclock' XEventsQueued
dtrace: script './all.d' matched 52377 probes
CPU FUNCTION
  0 -> XEventsQueued                       U
  0   -> _XEventsQueued                    U
  0     -> _X11TransBytesReadable          U
  0     <- _X11TransBytesReadable          U
  0     -> _X11TransSocketBytesReadable    U
  0     <- _X11TransSocketBytesreadable    U
  0     -> ioctl                           U
  0       -> ioctl                         K
  0         -> getf                        K
  0           -> set_active_fd             K
  0           <- set_active_fd             K
  0         <- getf                        K
  0         -> get_udatamodel              K
  0         <- get_udatamodel              K
...
  0         -> releasef                    K
  0           -> clear_active_fd           K
  0           <- clear_active_fd           K
  0           -> cv_broadcast              K
  0           <- cv_broadcast              K
  0         <- releasef                    K
  0       <- ioctl                         K
  0     <- ioctl                           U
  0   <- _XEventsQueued                    U
  0 <- XEventsQueued                       U
```

Progettista

Possibili strutture

Disegno ed implementazione di un SO

- Non esiste una regola generale per il disegno e l'implementazione di un SO
- La struttura interna dei SO varia ampiamente
- Occorre iniziare definendo gli obiettivi e le specifiche, che sono influenzati dall'hardware disponibile e dal tipo di sistema
- **Obiettivi utente e obiettivi del progettista**
  - Utenti – il SO deve essere conveniente da usare, facile da imparare, affidabile, sicuro e veloce
  - Sistema – il SO deve essere facile da progettare, implementare e gestire, così come, dovrebbe essere flessibile, affidabile, privo di errori ed efficiente

Disegno ed implementazione di un SO

- Principi da separare

  Politica: Cosa fare?
  Meccanismi: Come farlo?

- La separazione della politica dai meccanismi è un principio molto importante. Permette di ottenere la massima flessibilità se le decisioni di politica cambiano nel tempo

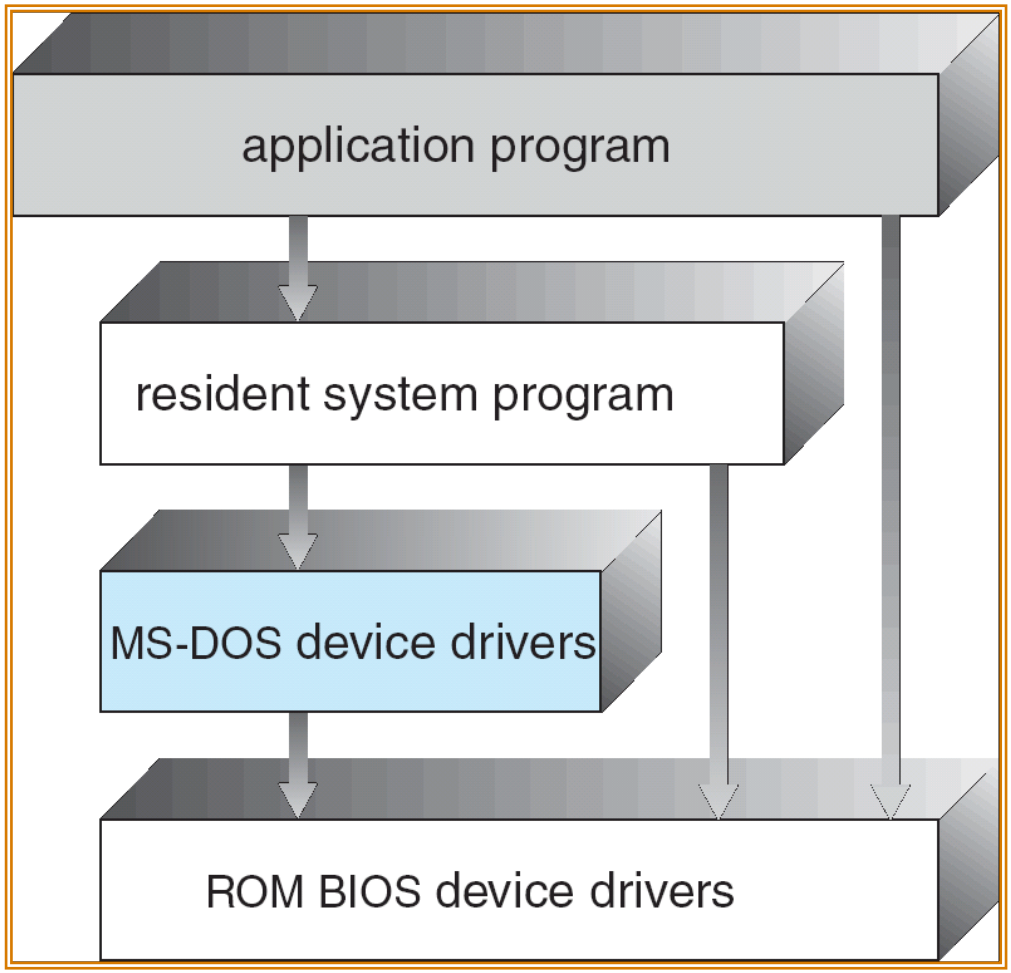
- Implementazione: Linguaggi ad alto livello

  - Porting su altre architetture

Struttura semplice

- MS-DOS – scritto per fornire la massima funzionalità possibile nel minimo spazio
  - Non è diviso in moduli
  - Anche se MS-DOS presenta una qualche struttura, le sue interfacce e i suoi livelli di funzionalità non sono ben separati
  - L'hardware per cui fu scritto non forniva supporto (no dual mode)

Struttura dei livelli in MS-DOS

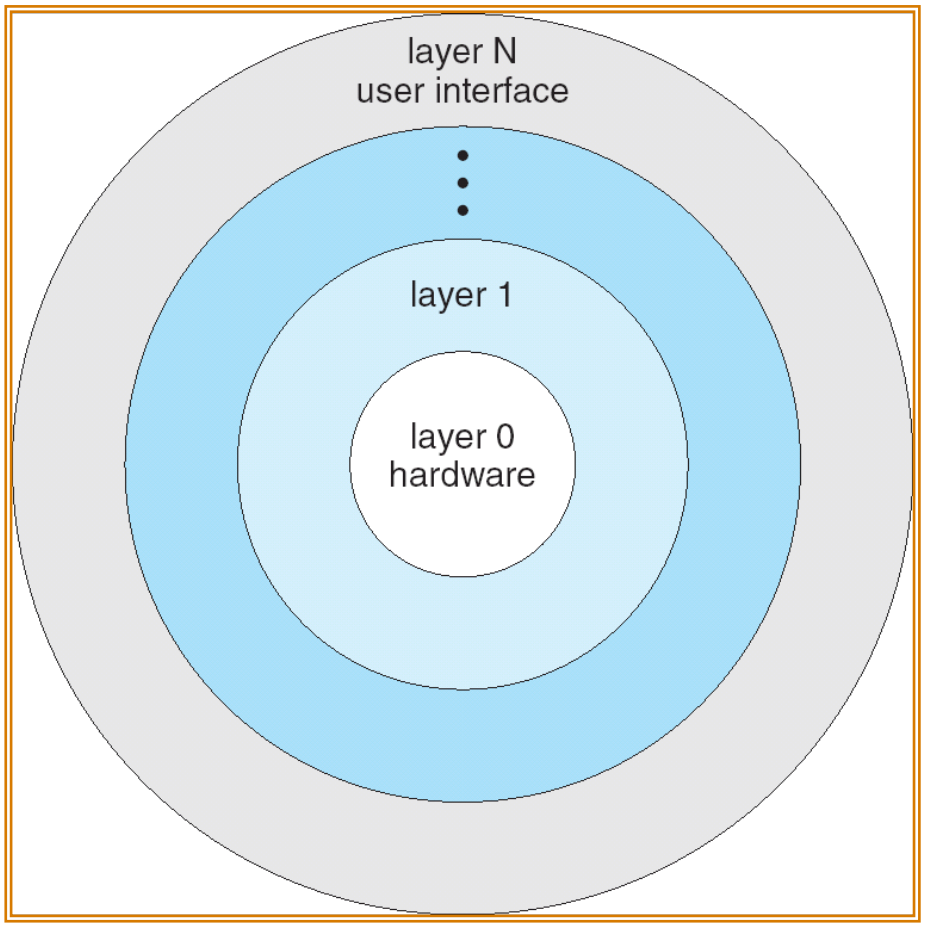


Approccio stratificato

- Il sistema operativo viene diviso in un certo numero di strati (livelli), ciascuno costruito al di sopra degli strati inferiori. Lo strato più in basso (strato 0), è l'hardware; quello più alto (strato N) è l'interfaccia utente

- Gli strati sono selezionati in modo tale che ciascuno di essi usa funzioni (operazioni) soltanto degli strati inferiori

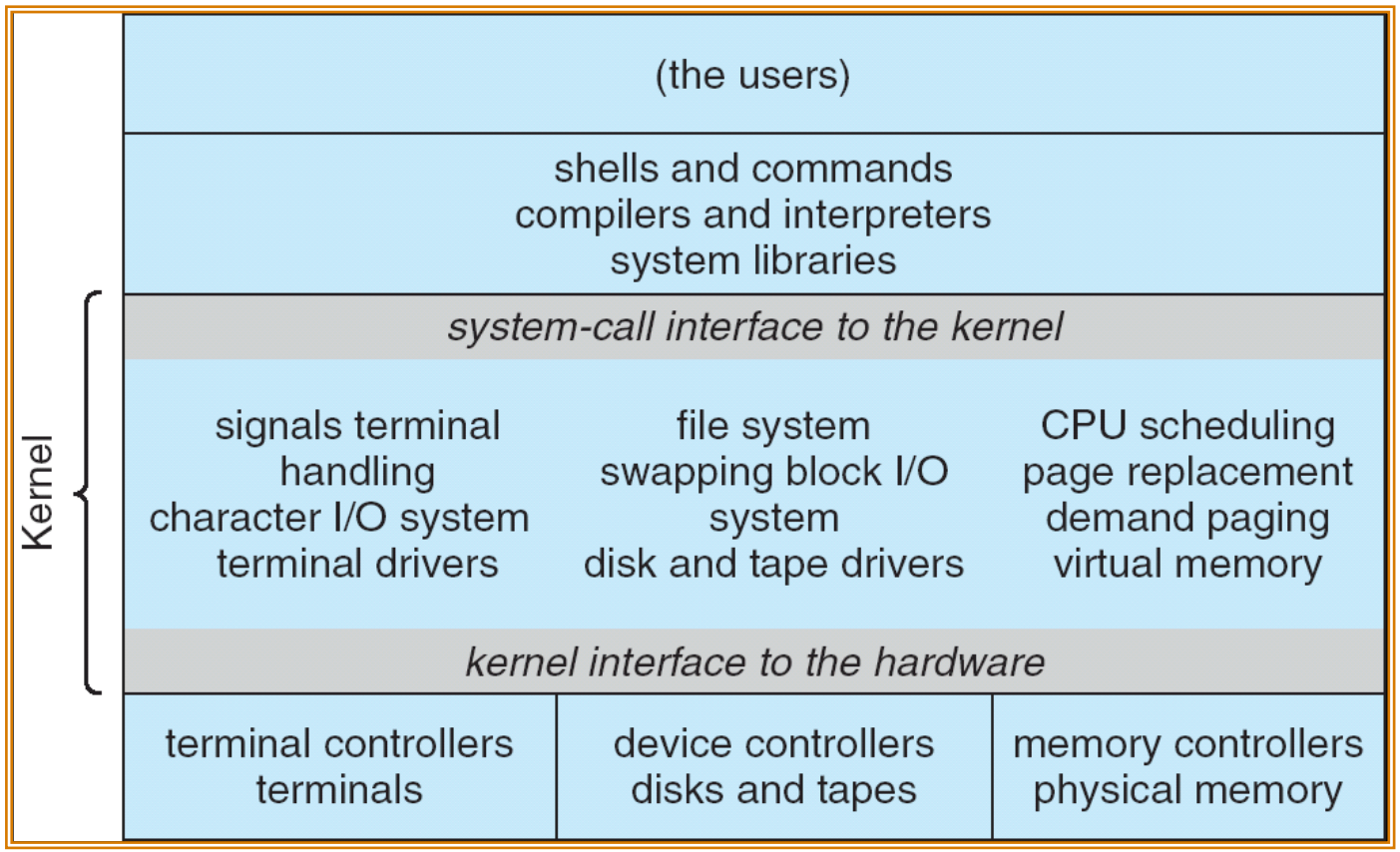
Sistema operativo a strati



Unix

- UNIX – limitato dalle funzionalità hardware, il sistema operativo Unix originale presentava struttura limitata. UNIX consiste di due parti separabili
  - Programmi di sistema
  - Kernel
    - Consiste di "tutto ciò che si trova al di sotto dell'interfaccia fornita dalle chiamate di sistema e al di sopra dell'hardware."
    - Fornisce il file system, la schedulazione della CPU, gestione della memoria, e altre funzioni del SO; un gran numero di funzioni per un singolo livello

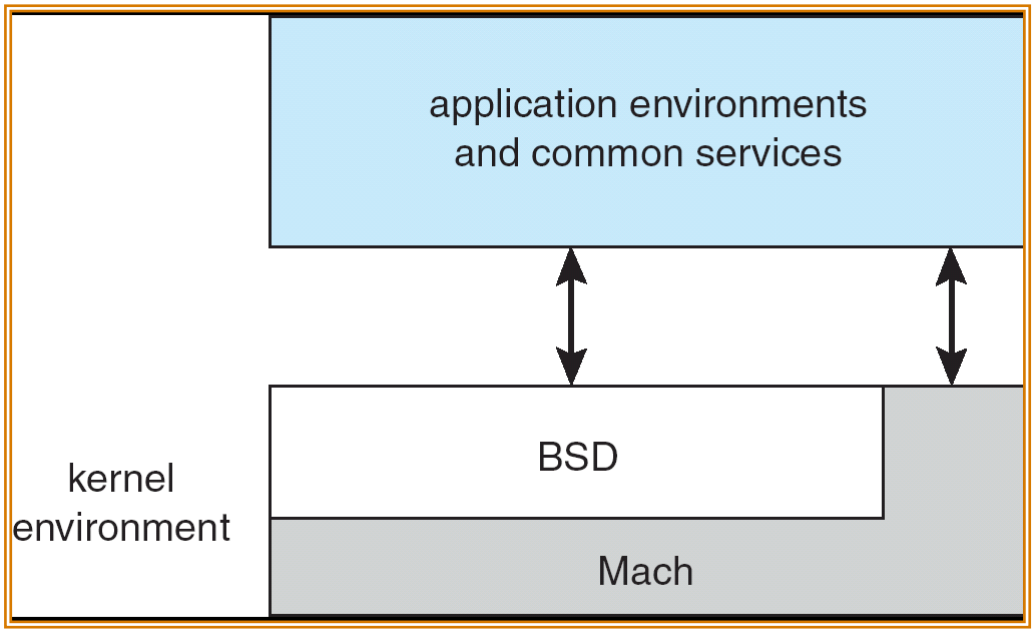
Struttura del sistema Unix



Microkernel

- Sposta molte delle funzionalità realizzate nel kernel in programmi che vengono eseguiti nello spazio utente
- Le comunicazioni tra gli utenti avvengono tramite il microkernel che gestisce lo scambio di messaggi
- Benefici:
  - Un microkernel è facile da estendere
  - Il SO è facile da portare su nuove archietture
  - Maggiore affidabilità (meno codice gira in kernel mode)
  - Maggiore sicurezza
- Riduzione delle prestazioni:
  - Maggiore overhead è richiesto per le comunicazioni tra processi utente e kernel

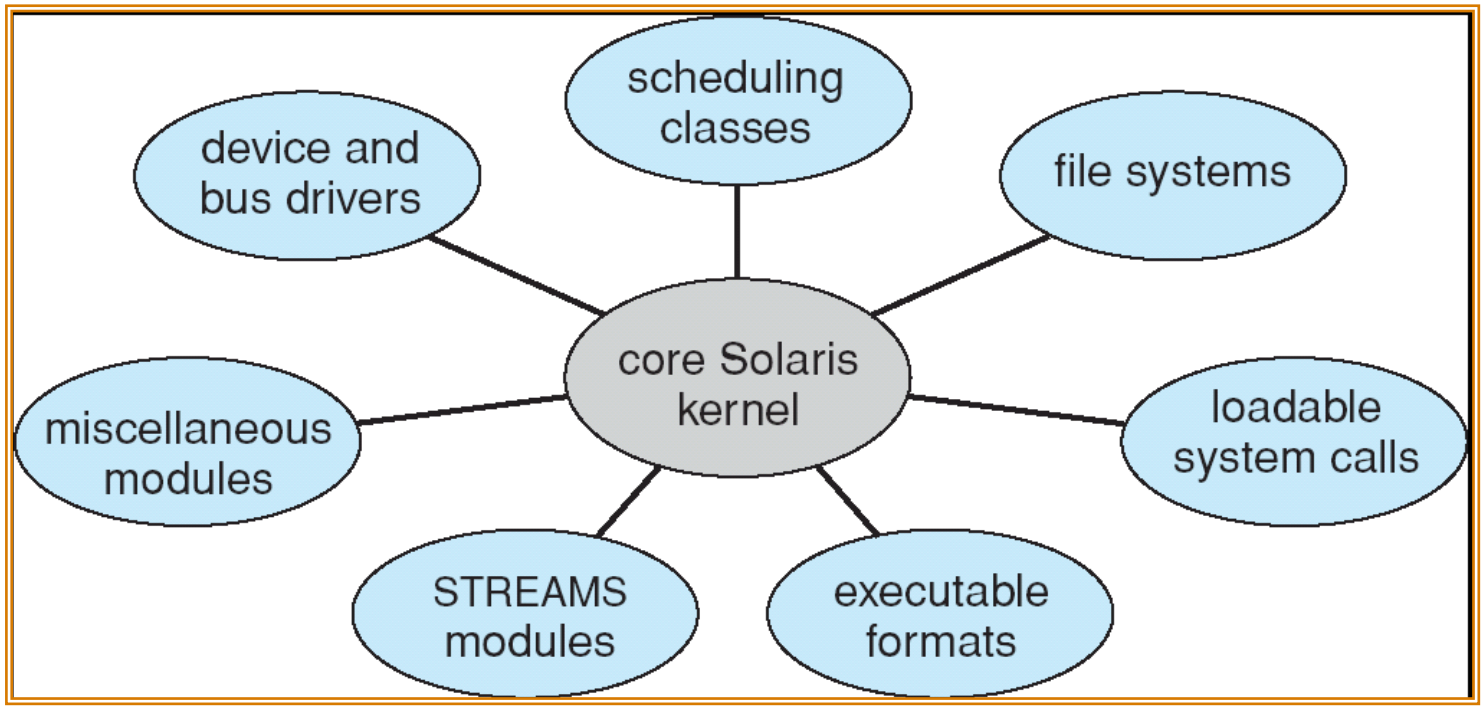
Struttura di Mac OS X



Moduli

- Molti sistemi operativi moderni implementano moduli kernel
  - Usano un approccio object-oriented
  - Ogni componente è separata
  - Ognuna di essa dialoga con le altre componenti attraverso interfacce
  - Ciascuna di esse può essere caricata quando necessaria all'interno del kernel
- L'approccio dei moduli è molto simile a quello stratificato (livelli) ma presenta maggiore flessibilità.

Solaris: Moduli



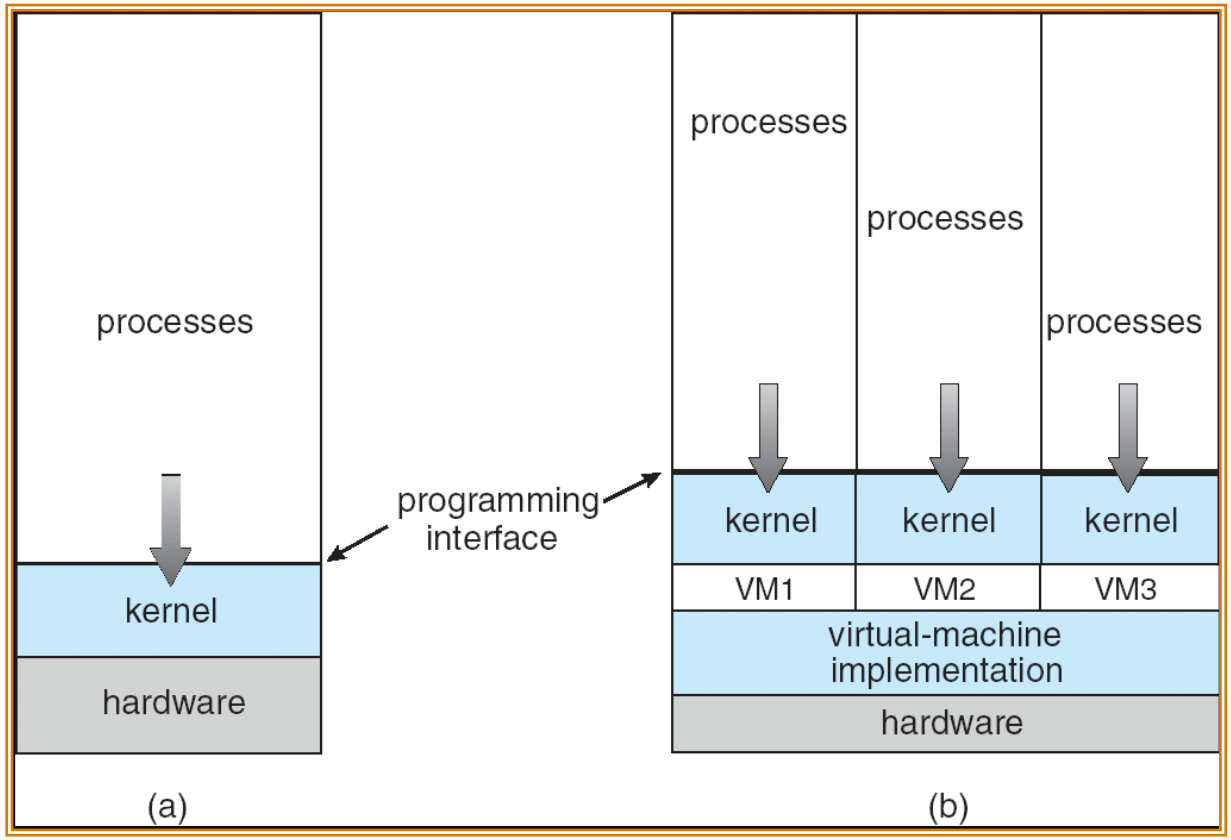
Macchine Virtuali

- Idea: realizzare da un'unica macchina reale (hardware) più **macchine virtuali**
- Una macchina virtuale fornisce un'interfaccia identica a quella offerta dalla sottostante macchina reale
- Il SO crea l'illusione di processi multipli, ciascuno in esecuzione sul proprio processore e all'interno della propria memoria (virtuale)

Macchine Virtuali

- Le risorse del computer fisico sono condivise per creare le macchine virtuali
  - La schedulazione della CPU fornisce agli utenti l'illusione di disporre di un proprio processore
  - Lo spooling ed il file system forniscono lettori di schede e stampanti virtuali
  - Un normale terminale di un sistema time-sharing rappresenta la console per l'operatore della macchina virtuale

Macchine Virtuali



Non-virtual Machine

Virtual Machine

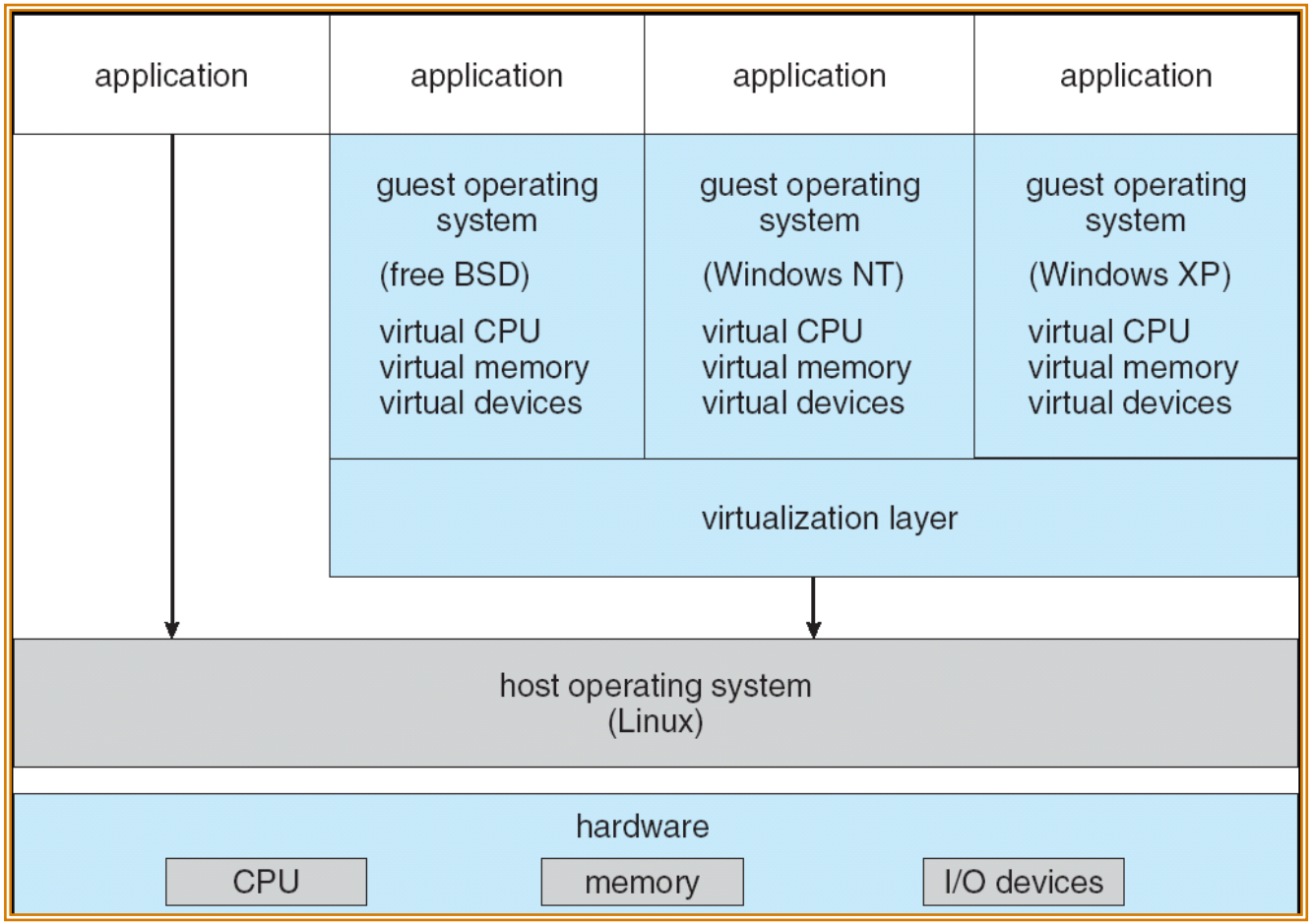
Macchine Virtuali

- Il concetto di macchina virtuale fornisce protezione completa delle risorse del sistema, poichè ciascuna macchina virtuale è isolata dalle altre macchine. Questo isolamento, d'altra parte, non permette la condivisione delle risorse.

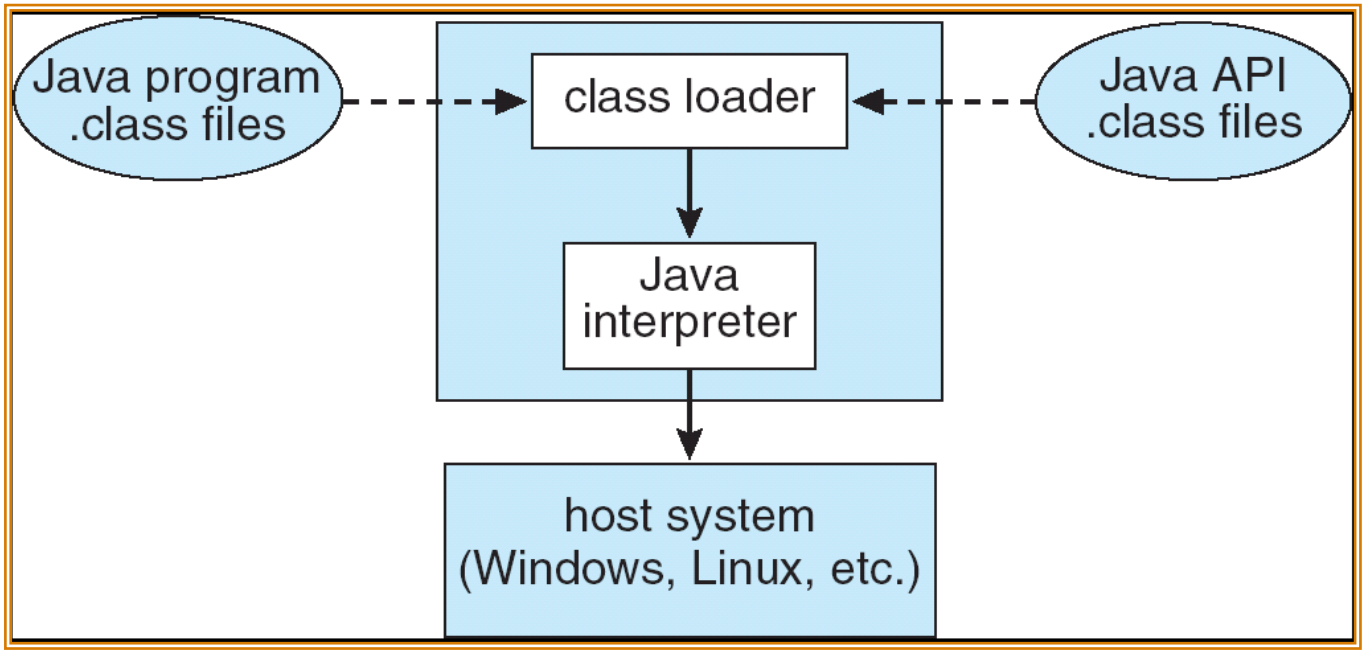
- Un sistema di macchine virtuali è uno strumento perfetto per la ricerca e lo sviluppo di nuovi sistemi operativi. Lo sviluppo del sistema viene fatto sulla macchina virtuale, invece che sulla macchina fisica. Pertanto, non genera conflitti con le normali operazioni del sistema

- Il concetto di macchina virtuale è difficile da implementare

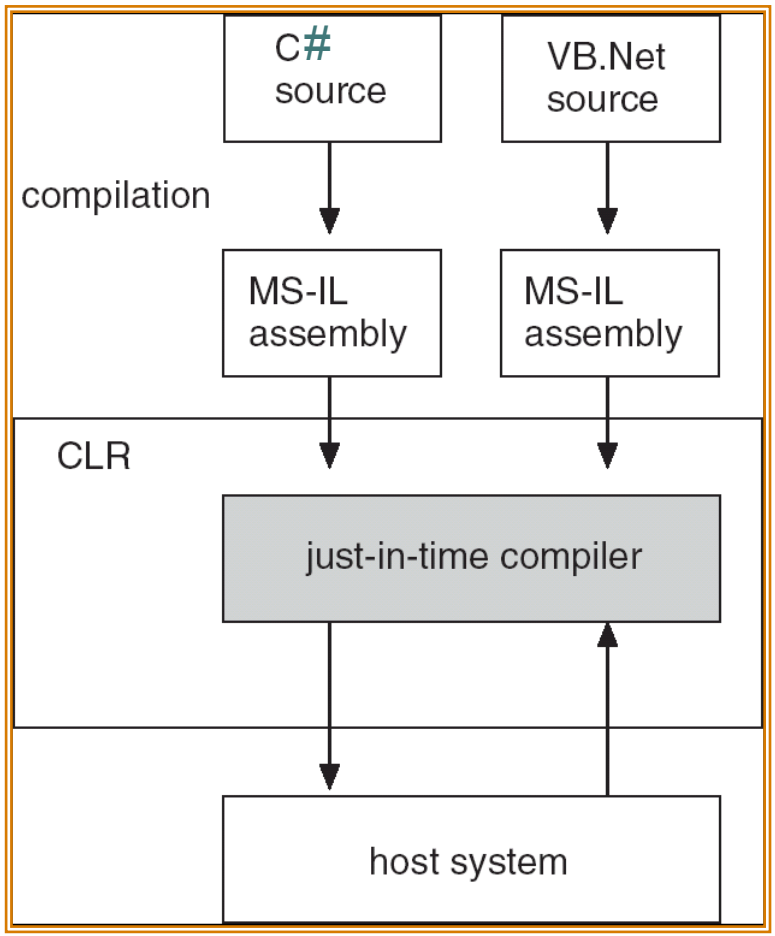
Architettura VMware



La Java Virtual Machine



Framework .NET



Generazione di un Sistema Operativo

- I SO sono progettati per essere eseguiti su diverse classi di macchine; quindi, devono essere configurati per ogni specifico computer
- Solitamente un programma di generazione di sistema SYSGEN ottiene informazioni circa la specifica configurazione dell'hardware sottostante (cpu, memoria …)
- Approcci alla configurazione
  - Modifica del codice sorgente e ricompilazione
  - Selezione del "sottoinsieme" utile (codice precompilato)
  - Selezione dinamica tramite tabella (cod. precompilato)

Boot del sistema

- Un sistema operativo deve essere presente nella macchina in modo tale che l'hardware ne possa avviare l'esecuzione
  - Una piccola porzione di codice – **bootstrap loader**, localizza il kernel, lo carica in memoria centrale, e ne avvia l' esecuzione
  - Qualche volta il processo è diviso in due fasi: un **blocco** di **boot** in una locazione fissa del disco contiene il bootstrap loader
  - Quando il computer viene accesso, l'esecuzione parte da una locazione fissa di memoria
    - Del firmware (ROM, EPROM) viene usato per caricare in memoria il blocco di boot