Strutture dei Sistemi Operativi

Componenti del sistema
Servizi di un sistema operativo
Struttura dei sistemi operativi

Gestione dei processi

- Un *processo* è un programma in esecuzione
  - Un processo ha bisogno di alcune risorse: tempo di CPU, memoria, file, dispositivi di I/O, per soddisfare ai compiti richiesti
- Il sistema operativo è responsabile delle seguenti attività
  - Creazione e cancellazione di processi
  - Sospensione e ripristino di processi
  - Mettere a disposizione meccanismi per:
    - Sincronizzazione di processi
    - Comunicazione fra processi

Gestione della memoria centrale

- la memoria è condivisa dalla CPU e dai dispositivi di I/O
  - è l'unico dispositivo di memorizzazione che la CPU può indirizzare direttamente
  - le istruzioni possono essere eseguite dalla CPU solo se si trovano effettivamente in memoria
  - per ovviare a questi vincoli per migliorare l'utilizzo della CPU e quindi le prestazioni complessive del sistema occorre poter gestire

    più programmi in memoria
  - esistono allora più schemi per la gestione della memoria, che ovviamente dipendono dall'hardware

Gestione della memoria secondaria

- Il sistema operativo è responsabile di
  - gestire lo spazio libero
  - allocare lo spazio
  - fare lo scheduling del disco
- Dato che l'uso del disco è molto frequente la gestione deve essere efficiente vi è stato perciò uno studio molto accurato per la ricerca degli algoritmi migliori

Gestione del sistema di I/O

- **Scopo**: nascondere l'hardware all'utente
  - sistema di buffer-caching
  - interfaccia generale per i driver dei dispositivi
  - driver per specifici dispositivi hardware
- Solo il driver conosce le caratteristiche dello specifico dispositivo a cui è assegnato

Gestione dei file

- Supporti + caratteristiche
  - nastri magnetici, dischi magnetici, dischi ottici

    +
  - velocità, capacità, velocità di trasferimento dati, metodi d'accesso (casuale o sequenziale)
- Il sistema operativo ha una visione logica uniforme del processo di memorizzazione dell'informazione:
  esiste un solo tipo di oggetto
- il file è la sola unità logica di memoria
  - creazione e cancellazione di file
  - creazione e cancellazione di directory
  - supporto di primitive per la manipolazione di file e directory
  - mapping dei file sulla memoria secondaria
  - backup dei file su dispositivi di memorizzazione stabile

Reti

- Un sistema distribuito è un insieme di unità che non condividono la memoria, i dispositivi periferici o un clock
  - Ogni unità dispone di una propria memoria locale
- Le unità comunicano tra loro tramite reti
  - Le comunicazioni avvengono tramite protocolli
- Dal punto di vista dell'utente, un sistema distribuito è un insieme di sistemi fisicamente separati organizzato in modo tale da apparire come un unico sistema coerente
  - Appare come una singola memoria principale ed un unico spazio di memoria di massa
  - Utile per file system distribuiti
- L'accesso ad una risorsa condivisa permette:
  - di accelerare il calcolo
  - di aumentare la disponibilità di dati
  - di incrementare l'affidabilità

Gestione delle protezioni

- Un sistema operativo con più utenti consente che i processi vengano eseguiti in

Concorrenza

- I processi devono essere protetti dalle attività di altri processi occorrono quindi

Meccanismi di autorizzazione

- **Protezione**: è il meccanismo che controlla l'accesso da parte di programmi, processi o utenti alle risorse di un sistema di calcolo

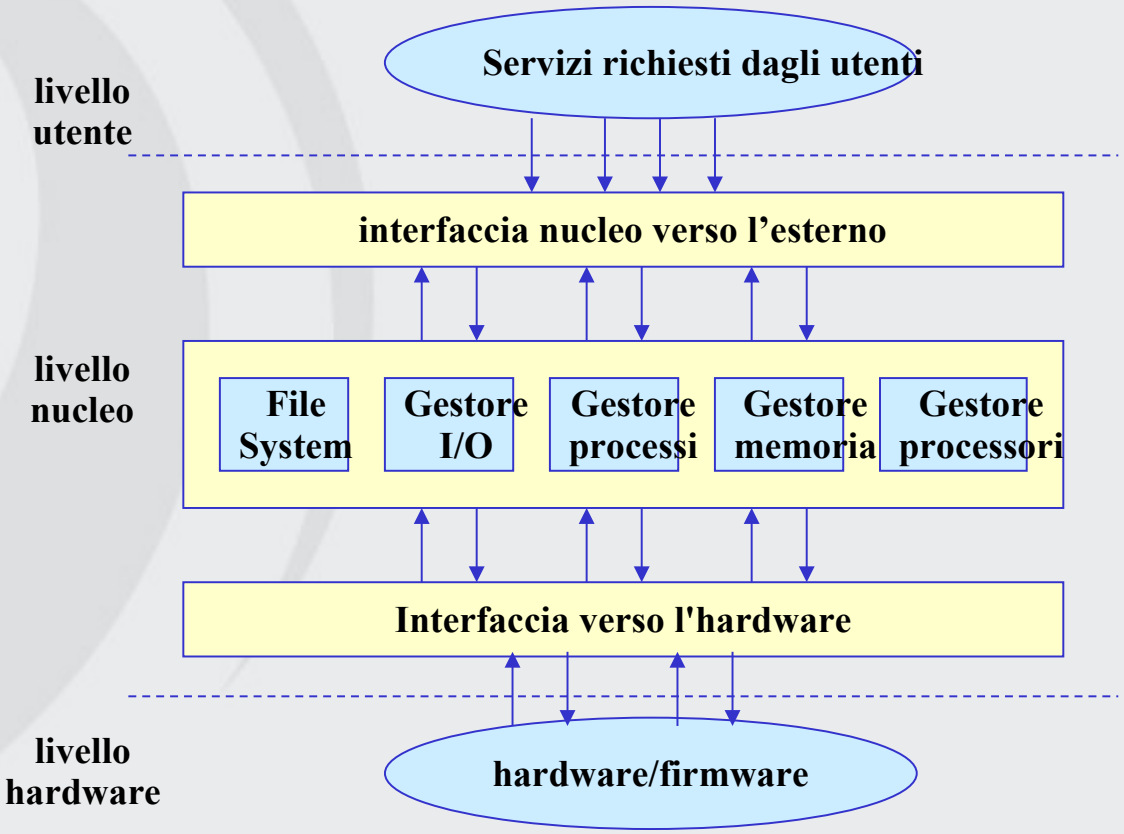
Interprete dei comandi

- L'interfaccia fra utente e sistema può avere aspetti diversi:
  - *Istruzioni di controllo*
  - *Shell*
  - *Interfaccia grafica*

Il suo compito è quello di accettare comandi da parte dell'utente

Il sistema operativo

- Un sistema operativo è costituito da **nucleo (kernel)** + **interfaccia verso l'utente**



System call

- L'interfaccia tra il Sistema Operativo e i programmi degli utenti è definita da un insieme di

  Istruzioni Estese
  ovvero
  System Call (chiamata di sistema)

  - Creano
  - Cancellano
  - Usano

  oggetti software gestiti dal sistema operativo

- La loro implementazione può essere diversa a seconda del sistema operativo

Tipi di chiamate di sistema

- Controllo dei processi
- Gestione dei file
- Gestione dei dispositivi
- Gestione dell'informazione
- Comunicazioni

Chiamate di sistema

- I programmi dell'utente comunicano con il sistema operativo e gli richiedono servizi per mezzo delle Chiamate di Sistema
- a ogni system call corrisponde una **Procedura di Libreria** (che il programma può invocare) che ha i seguenti compiti:
  - mette i parametri della system call in un sito predefinito (es. registro)
  - istanzia un'istruzione TRAP per mandare un interrupt al sistema operativo
  - nasconde i dettagli dell'istruzione TRAP
  - rende le system call come chiamate di procedura normali (es. da programmi in C)

Chiamate di sistema

UNIX	Win32	Descrizione
fork	CreateProcess	Crea un nuovo processo
waitpid	WaitForSingleObject	Aspetta che un processo termini
execve	[nessuna]	CreateProcess = fork + execve
exit	ExitProcess	Termina l'esecuzione
open	CreateFile	Crea un file o apre un file esistente
close	CloseHandle	Chiude un file
read	ReadFile	Legge dati da un file
write	WriteFile	Scrive dati in un file
lseek	SetFilePointer	Muove il file pointer
stat	GetFileAttributesEx	Restituisce gli attributi di un file
mkdir	CreateDirectory	Crea una nuova directory
rmdir	RemoveDirectory	Rimuove una directory vuota
link	CreateSymbiolicLink	Crea i collegamenti
unlink	Deletefile	Elimina un file
mount	[nessuna]	Win32 non supporta il mount
umount	[nessuna]	Win32 non supporta il mount
chdir	SetCurrentDirectory	Cambia la directory di lavoro
chmod	SetFileAttributes	Modifica gli attributi dei file
kill	TerminateProcess	Il comportamento differisce nei due S.O.
time	GetLocalTime	Restituisce data e ora corrente

Programmi di sistema

- I programmi di sistema offrono un ambiente per lo sviluppo e l'esecuzione dei programmi. Possono essere classificati in programmi per:
  - Gestione dei file
  - Informazioni di stato
  - Modifica dei file
  - Ambienti di sviluppo
  - Caricamento ed esecuzione dei programmi
  - Comunicazioni
  - Programmi applicativi
- Dal punto di vista dell'utente la maggior parte delle operazioni è vista come esecuzione di programmi, non chiamate di sistema

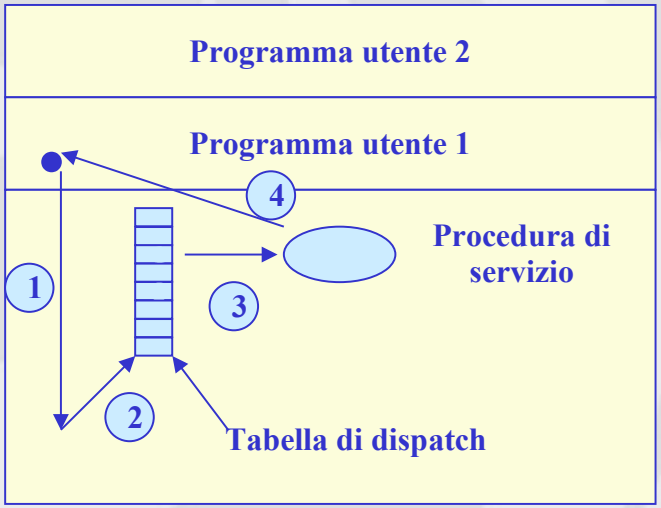
Struttura di un sistema operativo

- Sistemi monolitici
- Sistemi multilivello
- Modello client-server
- Macchine virtuali

Sistemi monolitici

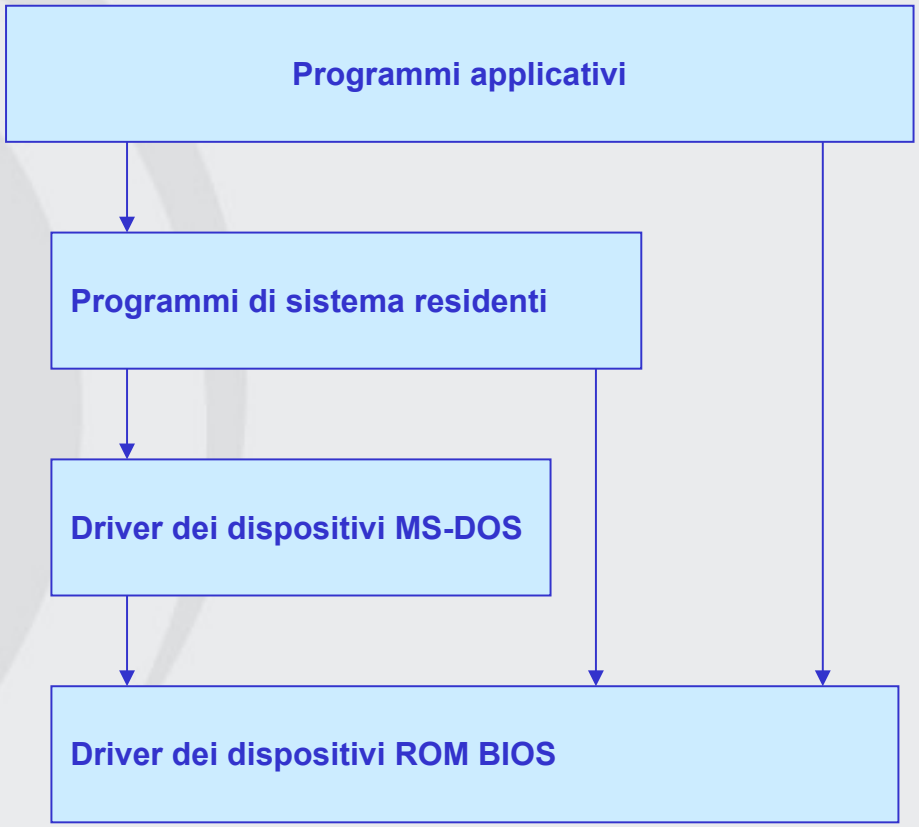
- il sistema operativo è costituito da una collezione di procedure ognuna delle quali può chiamare qualsiasi altra
- unica struttura presente sono le **system call** che comportano il salvataggio dei parametri e l'esecuzione di una trap speciale detta **kernel call** o **supervisor call**

Sistemi monolitici



- Come può essere realizzata una chiamata di sistema :
  1) il programma utente esegue una trap verso il nucleo;
  2) il sistema operativo determina il numero del servizio richiesto;
  3) il sistema operativo individua e chiama la procedura di servizio;
  4) viene restituito il controllo al programma utente

Struttura di MS-DOS

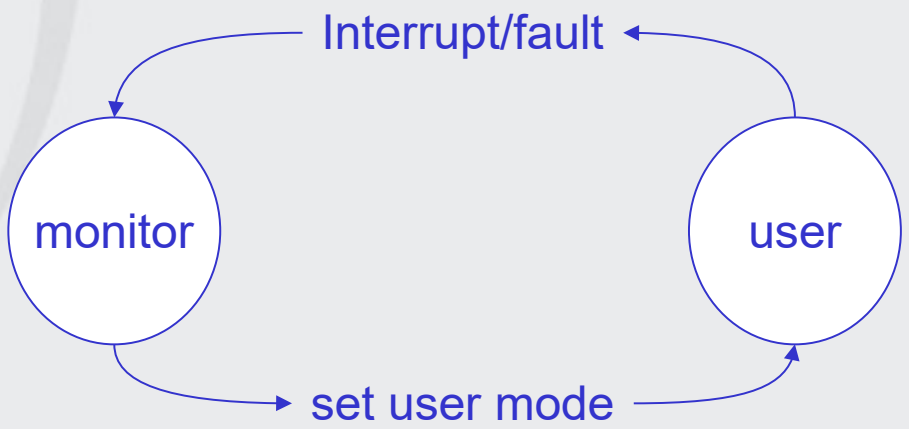


Duplice modo di funzionamento

- I sistemi che condividono risorse devono garantire che programmi non corretti non interferiscano con gli altri processi in esecuzione
- L'hardware (*mode bit*) permette al sistema operativo di operare in due modalità differenti
  - User mode – normale funzionamento dei programmi utente
  - Monitor mode (o kernel mode o system mode) – operazioni effettuate dal sistema operativo
- Quando avviene un interrupt si passa al kernel mode

Protezione dell'I/O

- Alcune istruzioni (privileged instructions) possono essere eseguite solo in kernel mode
- Tutte le istruzioni di I/O sono privileged instruction
- Occorre garantire che nessun programma possa essere eseguito in kernel mode

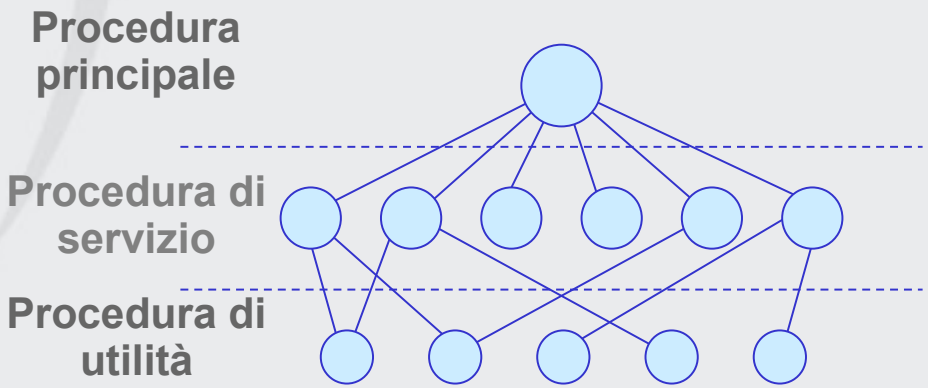


Struttura di UNIX

<table>
<tr><td colspan="3">Utenti</td></tr>
<tr><td colspan="3">Interprete dei comandi e comandi
Compilatori e interpreti
Librerie di sistema</td></tr>
<tr><td colspan="3">Chiamate di sistema</td></tr>
<tr><td>Segnali
Gestione dei terminali</td><td>File system
Sistema di I/O a blocchi</td><td>Scheduling della CPU
Memoria virtuale</td></tr>
<tr><td colspan="3">Interfaccia del kernel con l’hardware</td></tr>
<tr><td>Controllore di terminali
terminali</td><td>Controllore di dispositivi
Dischi e nastri</td><td>Controllore di memoria
Memoria fisica</td></tr>
</table>

Struttura a tre strati

1. Un programma principale che richiama la procedura di servizio richiesta
2. Un insieme di procedure di servizio che eseguono le chiamate di sistema
3. Un insieme di procedure di utilità che forniscono il supporto alle procedure di servizio



Sistemi multilivello

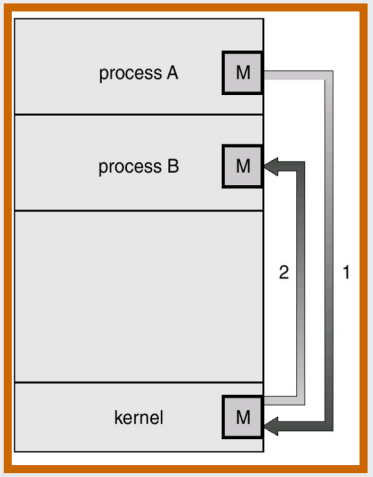
- il Sistema Operativo è organizzato in una gerarchia di livelli
- il 1° sistema con una struttura di questo tipo: **THE** realizzato alla Technische Hogelschool Eindhoven in Olanda da Dijkstra nel 1968 per il computer Electrologica X8 (32k di parole di 27 bit ed un tamburo di 512k parole usato per memorizzare parti di programma)
- THE ha 6 livelli

THE

Strato	Funzionalità
5	Operatore
4	Programmi Utente
3	Gestione I/O
2	Comunicazione processo-console
1	Gestione della memoria e del tamburo
0	Allocazione della CPU e multiprogrammazione

Struttura a microkernel

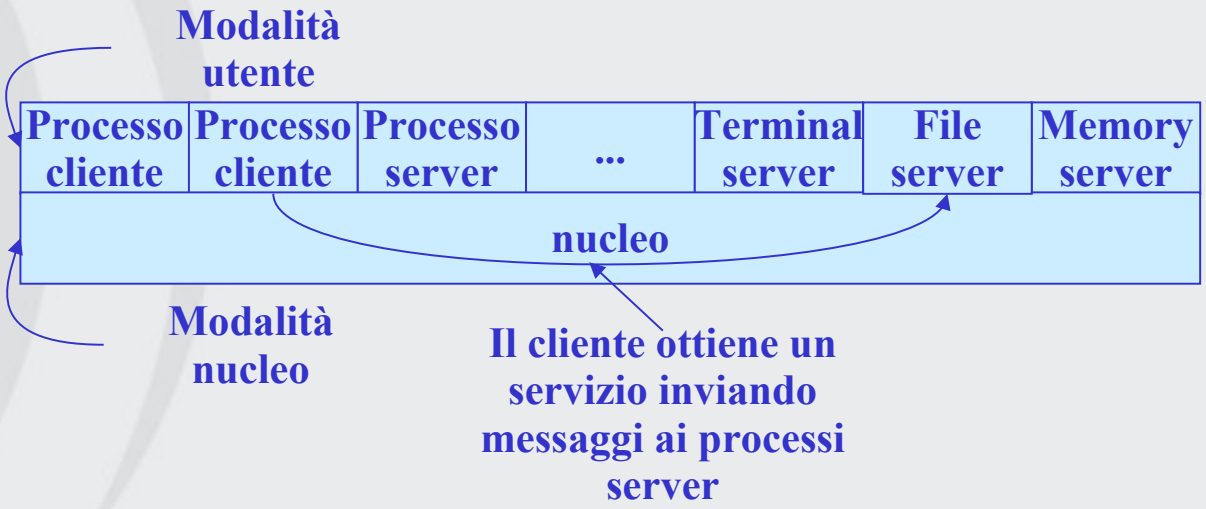
- Si sposta quanto possibile dal kernel allo spazio utente
- Le comunicazioni avvengono tramite messaggi
- Vantaggi:
  - Estendere un microkernel è più facile
  - Maggiore portabilità del sistema operativo
  - Più affidabilità (il codice critico nel kernel è minore)
  - Più sicuro
- Svantaggi:
  - Vi è un overhead dovuto alle comunicazioni fra spazio utente e kernel



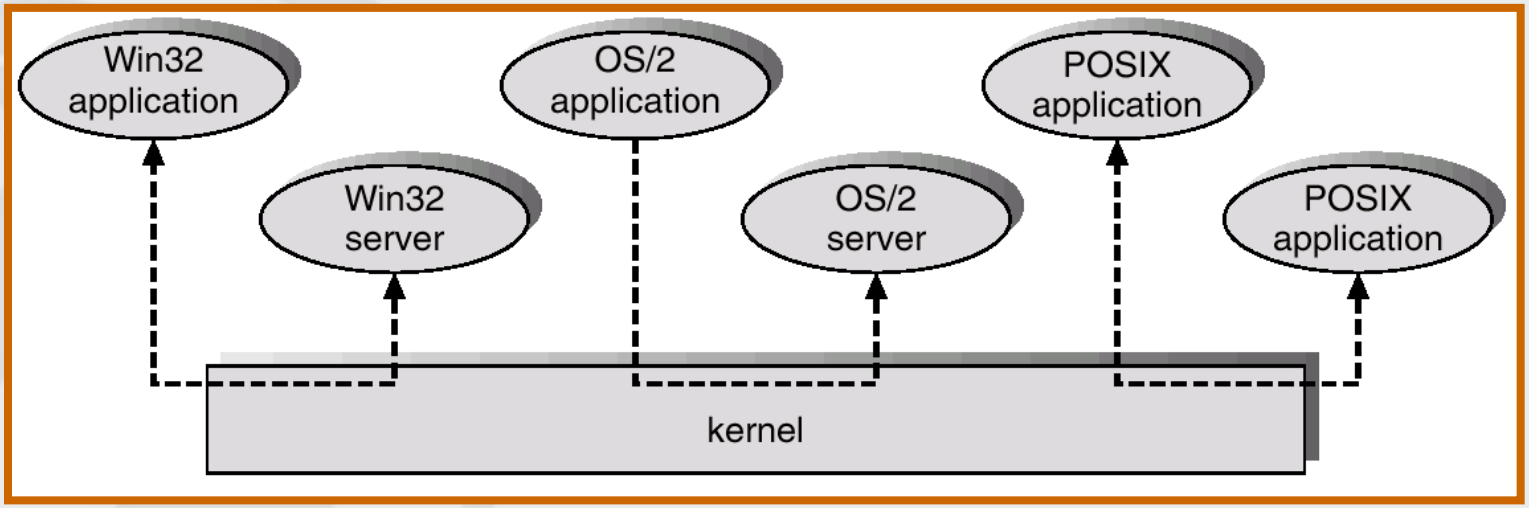
Modello client-server

- **Si basa sull'idea di portare il codice ai livelli superiori, lasciando un kernel minimo**
- **L'approccio è quello di implementare la maggior parte delle funzioni di sistema operativo nei processi utente**
  - **Processo Client**
    - Richiede un servizio
  - **Processo Server**
    - Gestisce le richieste del client
  - **Il kernel si occupa di gestire la comunicazione tra client e server**
- **Il sistema operativo viene suddiviso in più parti, piccole e maneggevoli, ognuna delle quali si riferisce ad un aspetto del sistema (memoria, file ...)**

Modello client-server

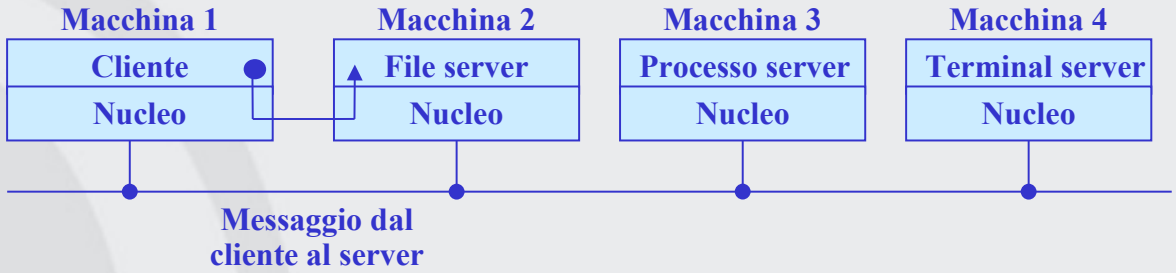


Struttura Client-Server di Windows NT



Vantaggi del modello client-server

- Vantaggi: adattabilità all'utilizzo in sistemi distribuiti



- Alcune funzioni di sistema operativo non sono eseguibili da programmi nello spazio utente
- Due soluzioni:
  1. processi server critici che vengono eseguiti in modalità kernel
  2. divisione tra:
     - meccanismo, minimo e implementato nel kernel
     - decisioni di strategia, lasciate ai processi server nello spazio utente

Moduli

- I più moderni sistemi operativi implementano un approccio modulare del kernel
  - Programmazione orientata agli oggetti
  - Ogni componente è separata
  - Comunica con le altre attraverso interfacce definite
  - Può essere caricato a sistema avviato
- Approccio simile ai livelli ma più flessibile

Moduli

```
osc@ubuntu:~/final-src-osc10e/ch2$ ls
Makefile hello.c simple.c
osc@ubuntu:~/final-src-osc10e/ch2$ make
make -C libmodules/4.4.0-87-generic/build M=/home/osc/.../ch2 modules
make[1]: Entering Directory '/usr/src/linux-headers-4.4.0-87-generic'
  CC [M]  /home/osc/final-src-osc10e/ch2/simple.o
  Building modules, stage 2
  MODPOST modules 1
  CC      /home/osc/final-src-osc10e/ch2/simple.mod.o
  LD [M]  /home/osc/final-src-osc10e/ch2/simple.ko
make[1]: Leaving Directory '/usr/src/linux-headers-4.4.0-87-generic'
osc@ubuntu:~/final-src-osc10e/ch2$ sudo insmod simple.ko
osc@ubuntu:~/final-src-osc10e/ch2$ sudo rmmod simple.ko
osc@ubuntu:~/final-src-osc10e/ch2$ tail /var/log/syslog
Mar 11 15:45:01 ubuntu kernel: [ … ] Loading Module
Mar 11 15:45:01 ubuntu kernel: [ … ] Removing Module
osc@ubuntu:~/final-src-osc10e/ch2$
```

Moduli

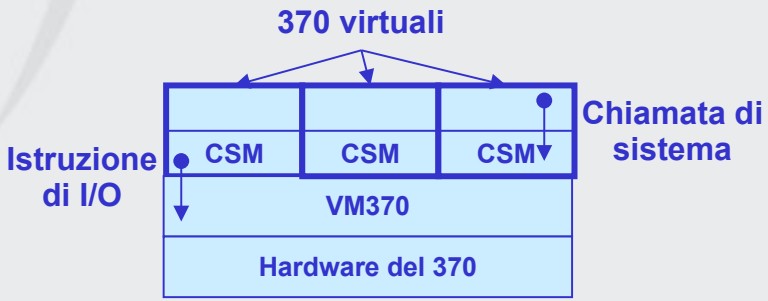
```
/**
 * simple.c
 *
 * A simple kernel module.
 *
 * To compile, run makefile by entering "make"
 *
 * Operating System Concepts - 10th Edition
 * Copyright John Wiley & Sons - 2018
 */
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

/* This function is called when the module is loaded. */
int simple_init(void)
{
    printk(KERN_INFO "Loading Module\n");
    return 0;
}

/* This function is called when the module is removed. */
void simple_exit(void) {
    printk(KERN_INFO "Removing Module\n");
}

/* Macros for registering module entry and exit points. */
module_init( simple_init );
module_exit( simple_exit );
MODULE_LICENSE("GPL"); MODULE_DESCRIPTION("Simple Module"); MODULE_AUTHOR("SGG");
```

Moduli

```
osc@ubuntu:~/final-src-osc10e/ch2$ sudo insmod hello.ko
osc@ubuntu:~/final-src-osc10e/ch2$ ls -l /proc/hello
-r—r--r-- 1 root root 0 Mar 11 15:59 /proc/hello
osc@ubuntu:~/final-src-osc10e/ch2$ cat /proc/hello
Hello World
osc@ubuntu:~/final-src-osc10e/ch2$ sudo rmmod simple.ko
osc@ubuntu:~/final-src-osc10e/ch2$ ls -l /proc/hello
ls: cannot access /proc/hello: No such file or directory
osc@ubuntu:~/final-src-osc10e/ch2$
```

Macchine virtuali

- Il sistema VM/370 (1972) si basa sulla separazione netta tra le funzioni di multiprogrammazione e di implementazione della macchina estesa
- Il Monitor della macchina virtuale gestisce la multiprogrammazione fornendo più macchine virtuali
- Ogni Macchina virtuale è una copia esatta dell'hardware
- Diverse macchine virtuali possono supportare sistemi operativi diversi
- CMS (Conversational Monitor System) - sistema interattivo per utenti timesharing



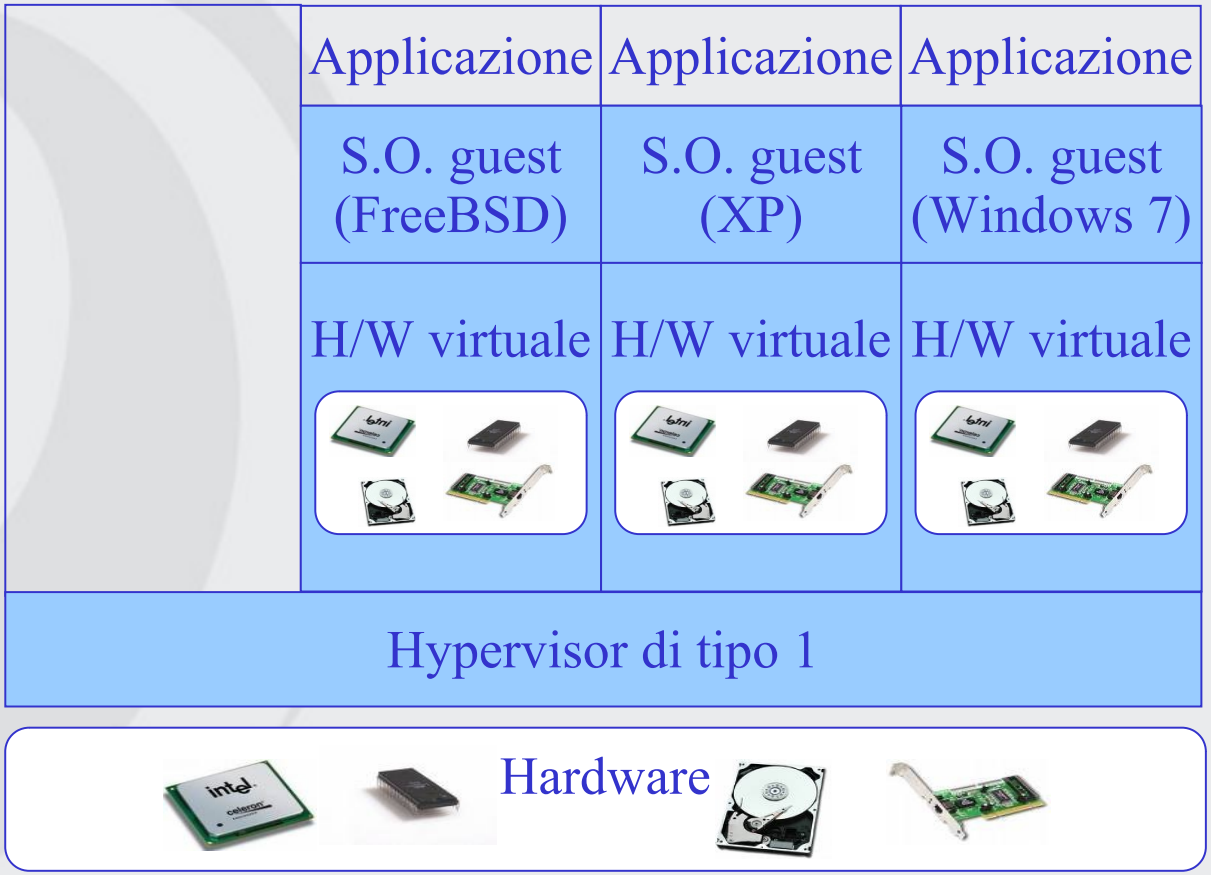
Macchine virtuali

- Vantaggi:
  - Ogni parte è più semplice, più flessibile e più facile da mantenere
  - Una macchina virtuale permette una completa protezione delle risorse di sistema
  - Rende facile lo sviluppo di un sistema operativo (le normali operazioni non vengono bloccate)
- Svantaggi:
  - L'isolamento delle macchine virtuali rende complessa la condivisione di risorse

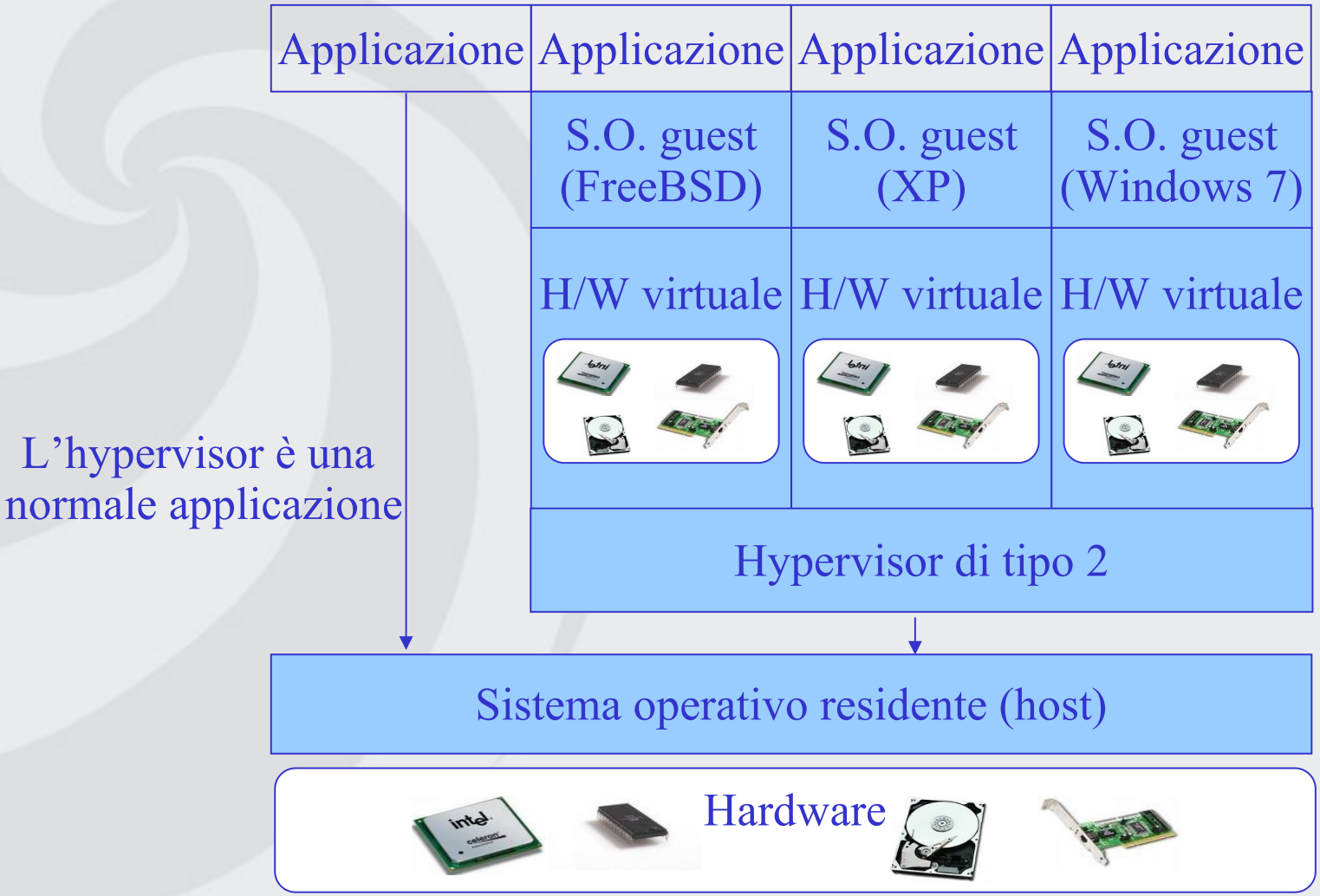
Modalità 8086

- Un approccio simile è utilizzato nei sistemi Windows con processore Pentium
- Il processore Pentium può funzionare in modalità virtuale 8086 In tale stato la macchina si comporta come un processore 8086 (indirizzamento a 16 bit e limite di memoria a 1M)
- Questa modalità viene utilizzata da WINDOWS, OS2, NT per eseguire vecchi programmi MS-DOS.
  Finché eseguono istruzioni normali lavorano direttamente sull'hardware della macchina, nel momento in cui fanno richieste al sistema operativo o istruzioni dirette di I/O, l'istruzione viene intercettata dalla macchina virtuale.

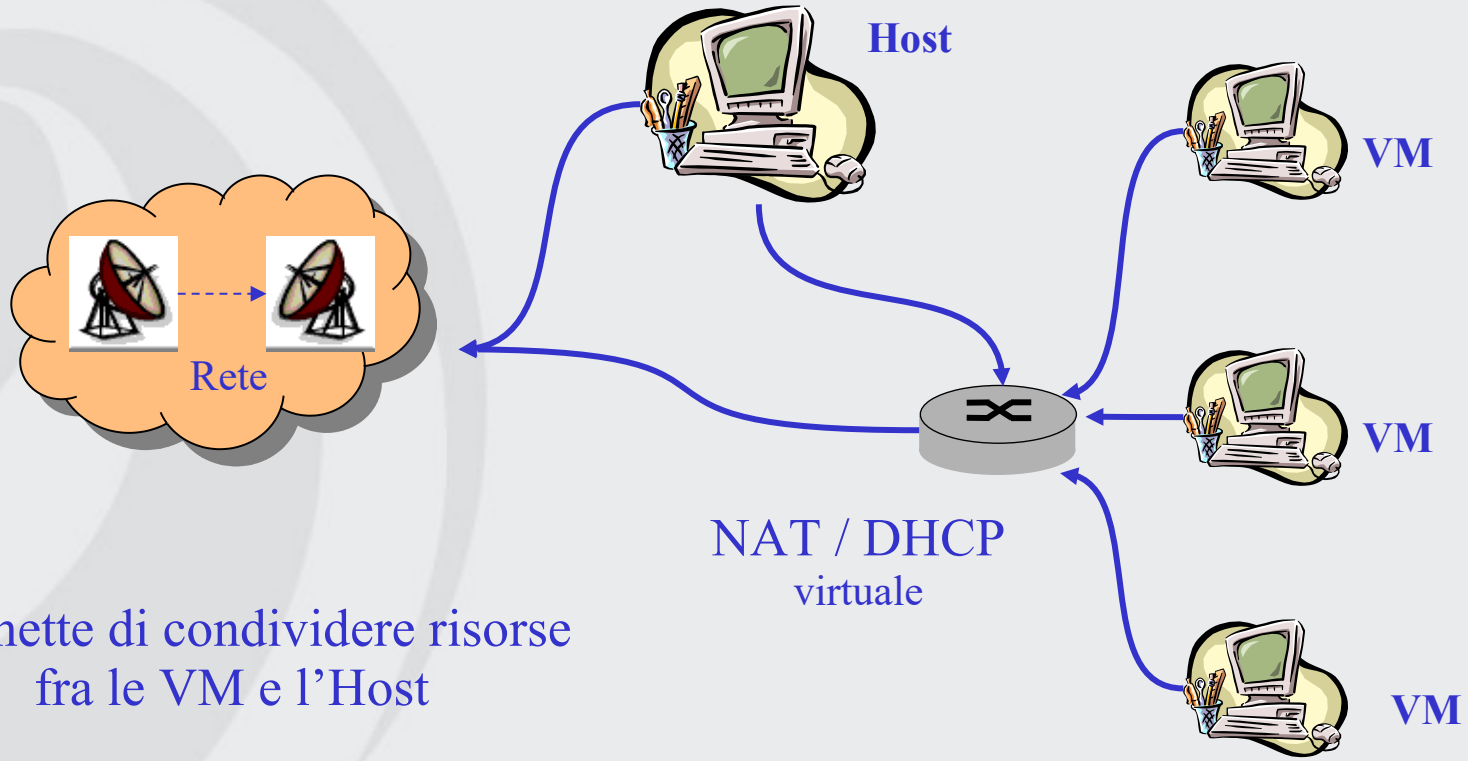
Hypervisor di tipo 1



Hypervisor di tipo 2

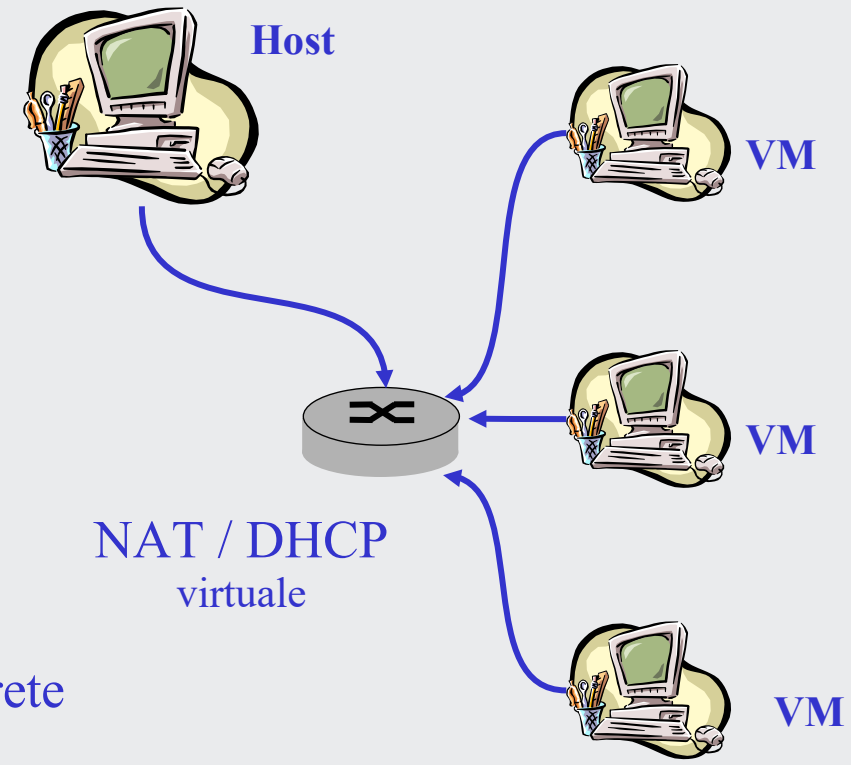


Rete virtuale



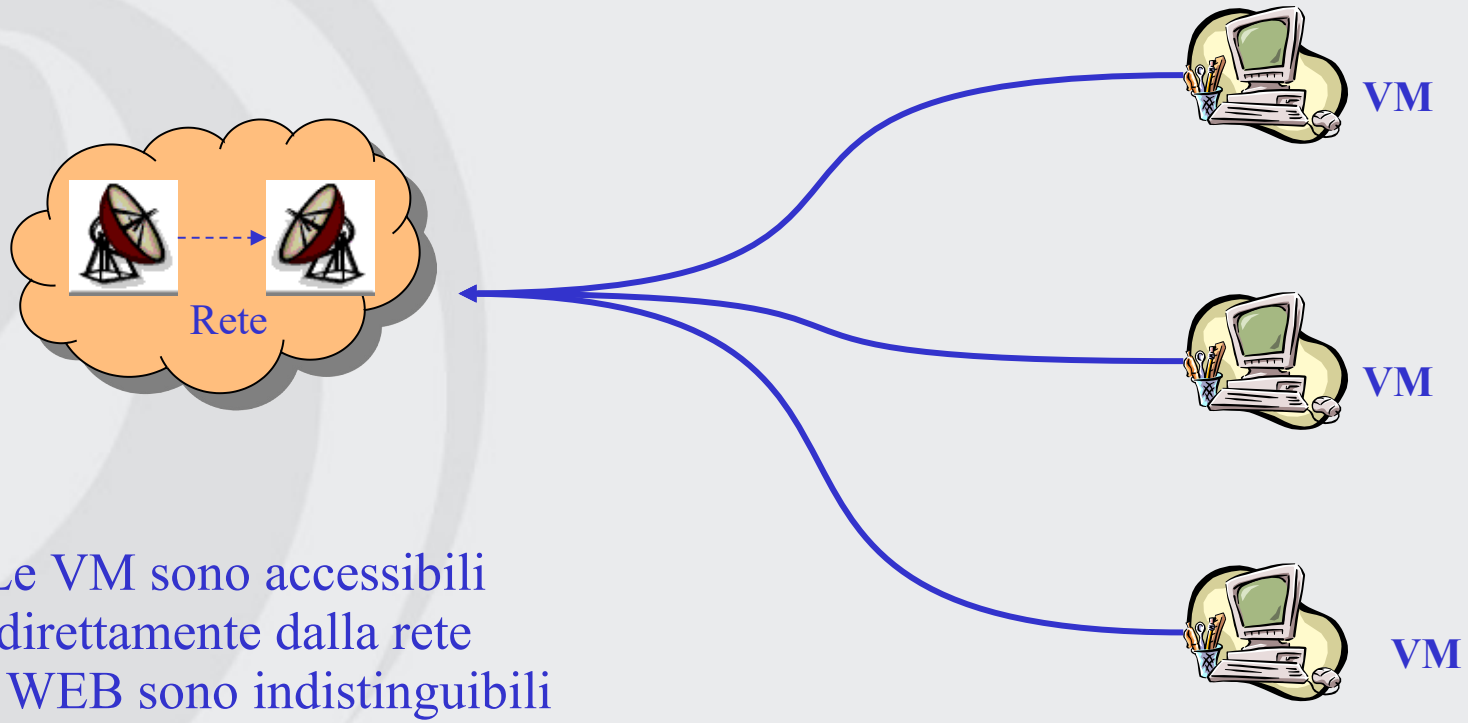
Permette di condividere risorse
fra le VM e l'Host

Rete virtuale



Permette di condividere risorse
fra le VM e l'Host senza accesso alla rete

Accesso diretto alla rete



Le VM sono accessibili direttamente dalla rete
Dal WEB sono indistinguibili rispetto a macchine reali

Perché usare una VM

- Possibilità di usare sistemi operativi diversi sulla stessa macchina
  - Minore costi hardware
  - Alternativa al dual boot – Sulla macchina sono disponibili più SO, ma solo uno alla volta
  - Utente tipico: sviluppatore web
- Più facile distribuzione del software:
  - la VM è già configurata e pronta all'uso
  - l'utente non deve preoccuparsi degli aggiornamenti
- Backup più semplici – Spesso non gestiti dall'utente

Esempi di virtualizzazione

- Bochs
- VMware Workstation 
- Xen
- Oracle VM VirtualBox 
- Hyper-V Microsoft

Requisiti per la virtualizzazione

- Le istruzioni vengono divise in tre gruppi:
  - Istruzioni privilegiate (generano una trap solo se si è in user mode)
  - Istruzioni di controllo (modificano le risorse di sistema, p.e. I/O)
  - Istruzioni il cui risultato dipende dalla configurazione del sistema
- Condizioni sufficienti perché una architettura possa essere virtualizzata (condizioni di Popek e Goldberg)
- Le istruzioni di controllo devono essere un sottoinsieme delle istruzioni privilegiate
  - Tutte le istruzioni che possono influenzare l'esecuzione del Virtual Machine Monitor devono generare una trap ed essere gestite dal VMM stesso
  - Le istruzioni non privilegiate possono essere eseguite nativamente

Tipi di virtualizzazione

- Emulazione
  - Permette l'esecuzione di un SO su una CPU completamente differente
  - Poco efficiente
- Virtualizzazione piena
  - Esegue copie di SO completi
- Paravirtualizzazione
  - Il sistema operativo ospitato deve essere modificato

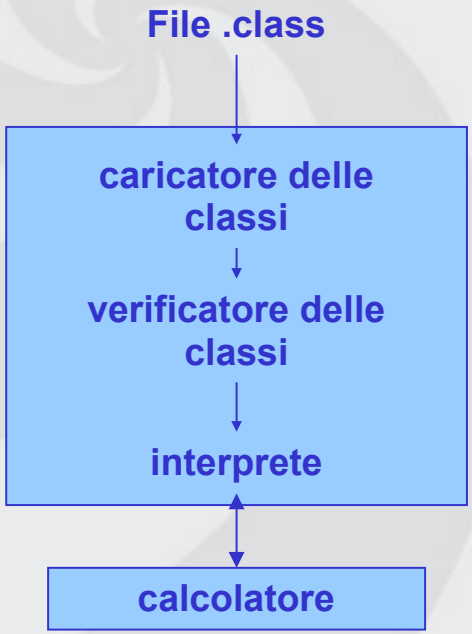
Binary rewriting

- Vengono esaminate le istruzioni del flusso del programma (a tempo di esecuzione)
- Vengono individuate le istruzioni privilegiate
- Vengono riscritte queste istruzioni con le versioni emulate
- Permette la virtualizzazione in spazio utente
- Più lento
- Si deve usare il caching delle locazioni di memoria
- Le prestazioni tipiche vanno dall'80% al 97% di una macchina non virtualizzata.
- Può essere implementato con i meccanismi dei debugger come i breakpoint

Paravirtualizzazione

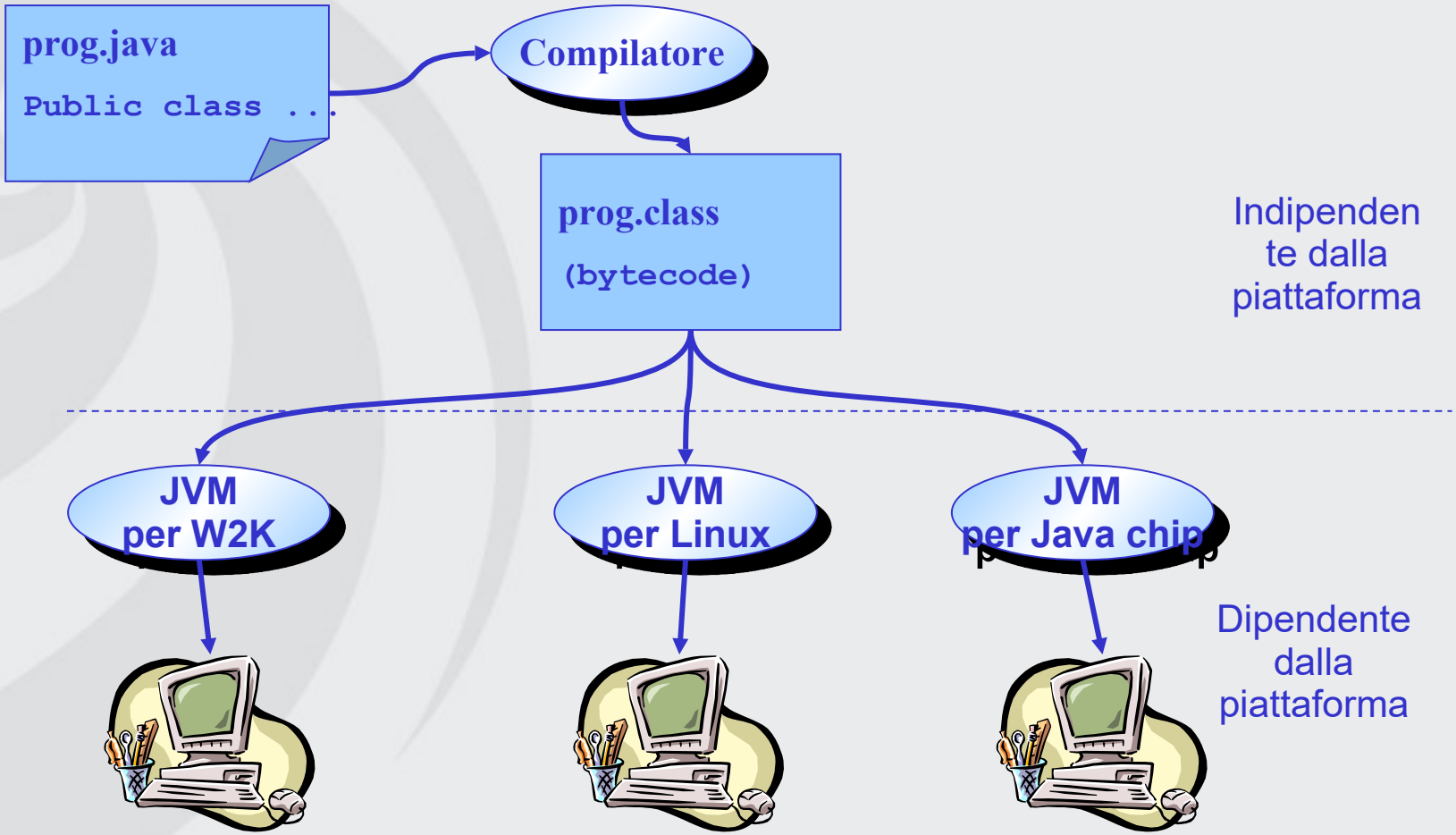
- Se un'istruzione del sistema guest genera una trap, questo ne deve gestire le conseguenze
- Il sistema guest deve essere modificato.
- Concettualmente simile al binary rewriting, ma il rewriting avviene a tempo di compilazione
- Quando un'applicazione genera una systemcall, questa viene intercetta dal SO (guest)
- Quando il SO guest prova ad eseguire istruzioni privilegiate, il VMM "intrappola" (traps) l'operazione e le esegue correttamente
- Quindi i SO guest effettuano Hypercall per interagire con le risorse del sistema.

Java Virtual Machine



- **È un calcolatore astratto che consiste di:**
  - **Un caricatore delle classi**
  - **Un verificatore delle classi**
    - (vengono controllati gli accessi alla memoria)
  - **Un interprete**
    - può essere un programma che interpreta gli elementi del bytecode uno alla volta o un compilatore istantaneo (just in time - JIT)
- **Gestisce inoltre la memoria procedendo in modo automatico alla sua ripulitura (Garbage Collection – ricupero della memoria non utilizzata)**

La piattaforma Java



Ambiente di sviluppo Java