Università di Roma Tor Vergata
Corso di Laurea triennale in Informatica

Sistemi operativi e reti

A.A. 2018-2019

Pietro Frasca

Lezione 3

Martedì 9-10-2018

Operazioni del sistema operativo

- I sistemi multiprogrammati richiedono che diversi programmi siano allocati contemporaneamente in memoria.
- Per avere diversi programmi in memoria allo stesso tempo è necessaria una qualche forma di ***gestione della memoria***.
- Inoltre, se alcuni programmi sono pronti per essere eseguiti allo stesso tempo, il sistema deve scegliere quale programma sarà eseguito per primo. Questa decisione è presa da una funzione del kernel, lo ***scheduling della CPU***.
- In un sistema time-sharing, il sistema operativo deve garantire un tempo di risposta brevi. Il metodo più comune per garantire tempi di risposta brevi si ottiene con la ***memoria virtuale***, una tecnica che permette l'esecuzione di un processo anche se non è completamente caricato in memoria. Il vantaggio principale della tecnica della memoria virtuale è che consente di eseguire programmi che sono più grandi della memoria fisica effettiva.
- Un sistema time-sharing deve anche fornire un file system. Il file system risiede su uno o più dischi e/o altri supporti di memorizzazione, di conseguenza, è necessaria la ***gestione della memoria secondaria***.

- Inoltre, un sistema time-sharing fornisce un meccanismo per proteggere le risorse da uso inappropriato.
- Per garantire un'esecuzione ordinata, il sistema deve fornire i meccanismi per ***la sincronizzazione e la comunicazione*** dei processi, e assicurare che questi non restino per sempre bloccati in una situazione di ***stallo***.
- Poiché il sistema operativo e i programmi utente condividono le risorse hardware e software del calcolatore, bisogna fare in modo che un errore in un programma utente non causi problemi ad altri programmi. Un programma erroneamente potrebbe modificare il codice o i dati di un altro programma, o anche il codice del sistema operativo stesso. Un sistema operativo progettato correttamente deve garantire che un programma non corretto, o volutamente dannoso, non possa modificare l'esecuzione di altri programmi.

Funzionamento dual-mode e multimode

- Per garantire la corretta esecuzione del sistema operativo, è necessario distinguere tra l'esecuzione di codice del sistema operativo e codice dei programmi utente.
- Per risolvere questo problema si utilizzano processori con un supporto hardware che permetta di differenziare tra i vari modi di esecuzione.
- Sono necessarie almeno due modalità di funzionamento: la ***modalità kernel*** (detta anche modalità privilegiata, di sistema o supervisore) e ***modalità utente***.
- Un bit, detto ***bit di modalità***, è presente nella CPU per stabilire la modalità di funzionamento corrente: kernel (0) o utente (1).
- Quando la CPU esegue un processo utente, il sistema è in modalità utente. Tuttavia, quando un'applicazione utente richiede un servizio dal sistema operativo, tramite una chiamata di sistema, il sistema deve passare da utente a modalità kernel.

- Questa tecnica di protezione si ottiene suddividendo le istruzioni macchina in istruzioni privilegiate e istruzioni non privilegiate.
- L'hardware della CPU consente di eseguire le istruzioni privilegiate solo in modalità kernel. Se si tenta di eseguire un'istruzione privilegiata in modalità utente, la CPU non esegue l'istruzione, ma piuttosto la considera come illegale e genera un'eccezione che sarà gestita dal sistema operativo.
- Ogni CPU ha il suo set di istruzioni privilegiate. Ad esempio, l'istruzione per passare alla modalità kernel è un'istruzione privilegiata. Alcuni altri tipi di istruzioni privilegiate sono le istruzioni per il controllo dell'I/O, per la gestione del timer e dell'interrupt.
- Il concetto di modalità può essere esteso oltre due modalità (nel qual caso la CPU utilizza più di un bit per impostare e testare la modalità).
- La mancanza di una modalità hardware dual-mode può causare gravi problemi in un sistema operativo.

- Un programma utente che esegue istruzioni errate può causare un crash del sistema operativo, scrivendo, ad esempio, in locazioni di memoria appartenenti al codice o ai dati del sistema operativo.
- I sistemi operativi come Microsoft Windows , Unix e Linux sono scritti per processori che utilizzano la funzionalità dual-mode e anche multi-mode.

Timer

- E' necessario assicurarsi che il sistema operativo mantenga il controllo sulla CPU. Non si deve permettere che un programma utente resti in esecuzione in un ciclo infinito o non restituisca il controllo del processore al sistema operativo.
- Per risolvere questo problema, si utilizza un timer. Un timer può essere impostato per generare un'interruzione dopo un determinato periodo. Prima di passare il controllo al programma utente, il sistema operativo imposta il timer affinché generi un interruzione allo scadere di un certo periodo. Quando il timer genera l'interruzione, il controllo della CPU ritorna automaticamente al sistema operativo.

- Chiaramente, le istruzioni che modificano il valore del timer sono privilegiate.

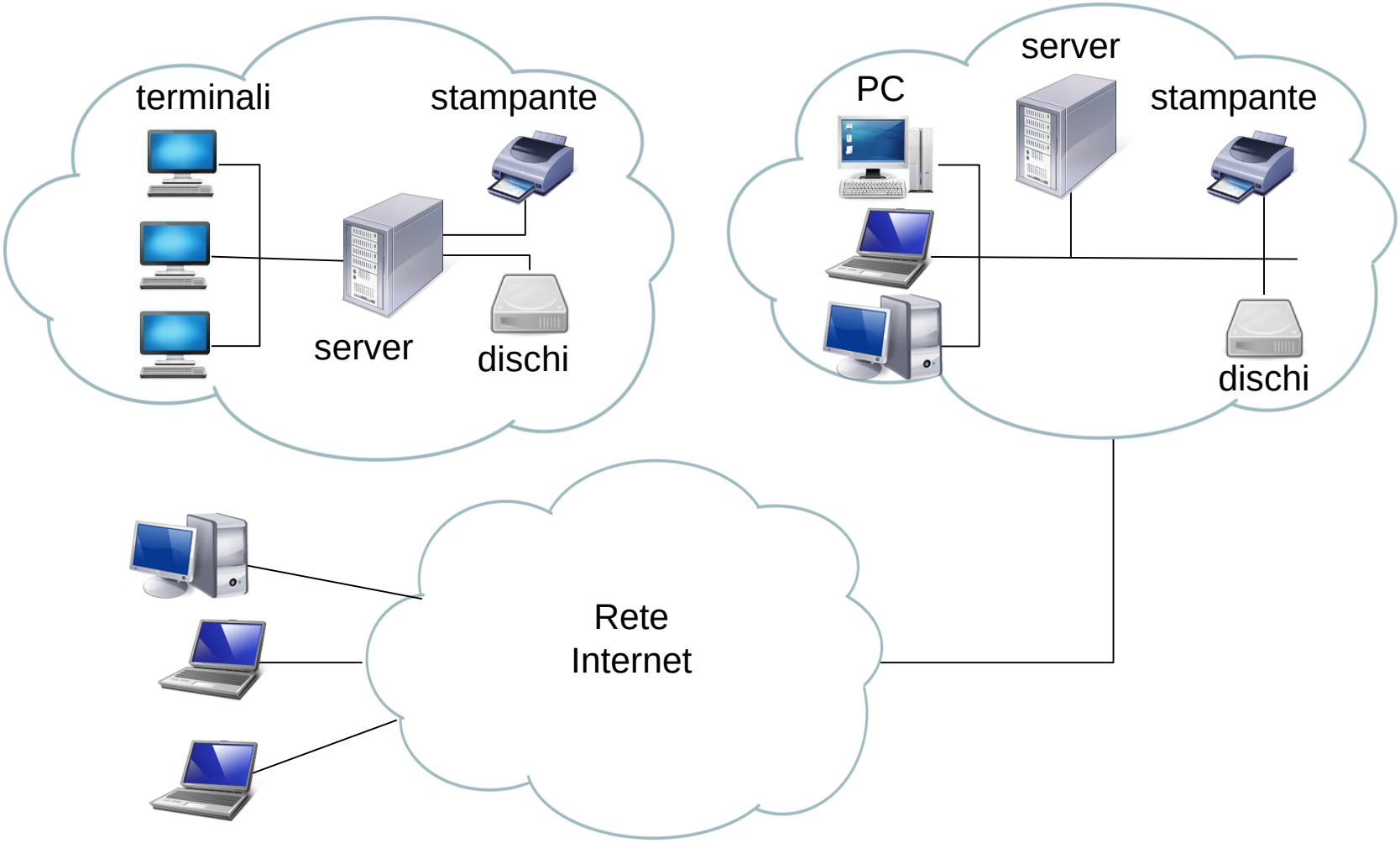
Ambienti di elaborazione

- Finora, abbiamo brevemente descritto alcune tipologie di sistemi operativi. Passiamo ora a spiegare come i sistemi operativi sono utilizzati in vari ambienti di elaborazione.

Computing tradizionale

- Solo pochi anni fa, un ambiente di elaborazione consisteva in un certo numero di PC collegati in rete, con un server che forniva servizi di condivisione di file e di stampa. L'accesso remoto era lento, limitato alla connessione tramite linea telefonica commutata con modem a 56 Kbit/s. Terminali collegati ai mainframe erano spesso usati in molte società e università.
- Con l'avvento della rete Internet, la tendenza attuale è di fornire varie modalità di accesso a questi ambienti informatici.
- Grazie alle tecnologie web e all'incremento della larghezza di banda WAN, le aziende forniscono portali web per accedere ai propri server interni.
- Computer desktop o portatili hanno sostituito i vecchi terminali.

- I computer mobili possono anche connettersi a reti wireless e alle reti dati cellulari per utilizzare il portale web della società.



Computing Mobile

- Il mobile computing si riferisce all'elaborazione su tablet e smartphone. Negli ultimi anni, tuttavia, la potenza di calcolo dei dispositivi mobili è talmente aumentata, in termini di funzionalità e prestazioni, tanto da rendere difficile la distinzione tra computer portatili e tablet.
- Oggi, i sistemi mobili sono utilizzati non solo per la posta elettronica e la navigazione su web ma anche per molte altre funzioni come la riproduzione di musica e video e la lettura di libri digitali.
- Molti sviluppatori ora realizzano applicazioni basate sulle caratteristiche integrate dei dispositivi mobili, come il sistema di posizionamento globale (GPS), accelerometri, giroscopi e altri sensori.
- Per fornire l'accesso ai servizi on-line, i dispositivi mobili in genere utilizzano sia le reti wireless wi-fi (IEEE 802.11) che le reti dati dei cellulari. La capacità di memoria e la velocità di elaborazione di dispositivi mobili, tuttavia, sono più limitati rispetto a quelle dei desktop.

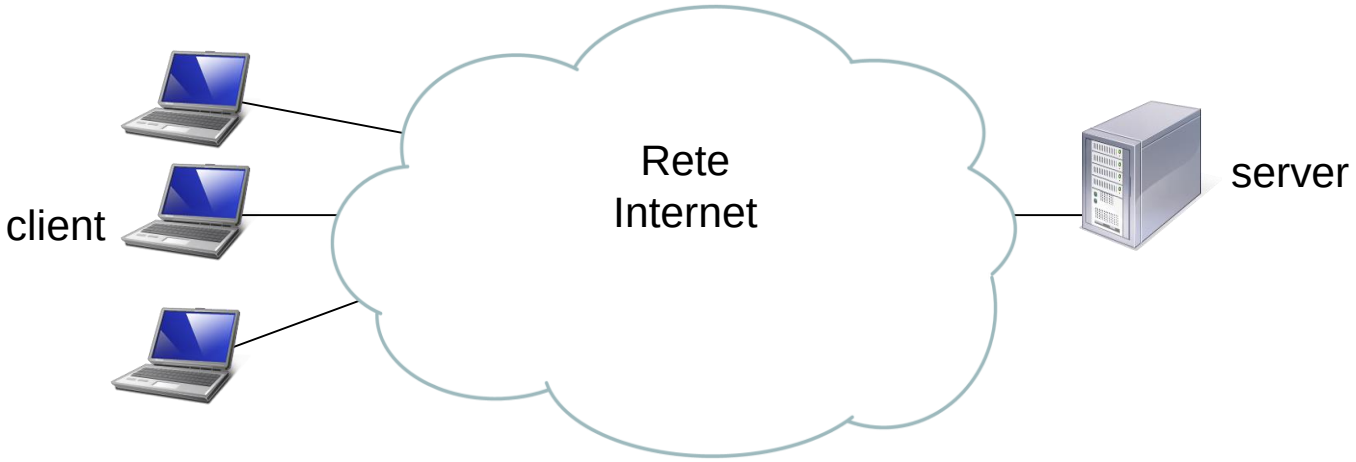
Sistemi paralleli e distribuiti

- I ***sistemi paralleli*** sono dotati di più CPU. In questi sistemi più processi potrebbero essere realmente eseguiti contemporaneamente.
- Il SO deve risolvere vari problemi legati all'esecuzione in parallelo dei programmi, nel caso che essi accedono simultaneamente agli stessi dati.
- Deve prevenire possibili condizioni di inconsistenza sulle strutture dati del sistema, che si potrebbero verificare se queste fossero modificate contemporaneamente da funzioni di SO eseguite su diverse CPU.
- Lo sviluppo delle reti ha portato alla realizzazione di moduli di SO per la gestione delle schede di rete e di vari servizi che hanno permesso ai sistemi collegati in rete di condividere dati e informazioni in vario modo come ad esempio tramite il web, e alla possibilità di realizzare applicazioni distribuite.

- I SO per calcolatori collegati in rete sono generalmente classificati in due categorie:
  - ***Sistemi operativi di rete***: ogni nodo della rete dispone di un proprio sistema operativo (non necessariamente uguale). I sistemi cooperano tra di loro, per esempio in applicazioni web a n-strati.
  - ***Sistemi operativi distribuiti***: in ogni nodo è installato lo stesso SO. L'utente vede l'intero sistema come un unico calcolatore. Il SO deve provvedere a bilanciare il carico di lavoro distribuendolo opportunamente sui vari nodi del sistema. I nodi condividono le risorse. Un esempio è dato dai sistemi **cluster**.

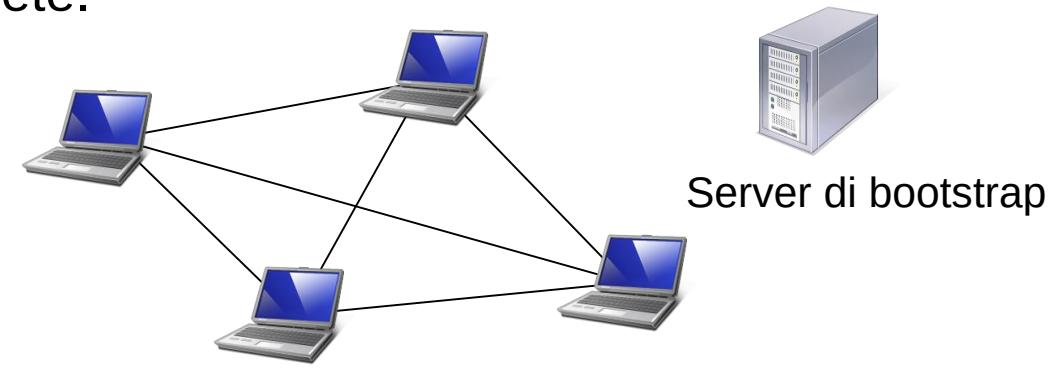
Client-Server

- Da quando i PC sono diventati più veloci, più potenti e meno costosi, i progettisti hanno abbandonato l'architettura di sistema centralizzato. Terminali collegati a sistemi centralizzati sono ora sostituiti da PC e dispositivi mobili dotati d'interfaccia grafica che si connettono ai server prevalentemente attraverso un'applicazione web.
- Di conseguenza, molti dei sistemi attuali funzionano come sistemi server per soddisfare le richieste generate da sistemi client. Questa forma di sistema, detto sistema ***client/server***, ha la struttura illustrata nella figura seguente.



Computing Peer- to-Peer

- Un altro modello di sistema è il ***peer- to-peer (P2P)***. In questo schema, tutti i nodi all'interno del sistema sono considerati pari, ciascuno può funzionare sia da client che da server. Ogni nodo può sia offrire che richiedere servizi.
- I sistemi Peer-to-peer offrono un vantaggio rispetto ai sistemi client/server. In un sistema client/server, il server è l'unico nodo a fornire servizi, mentre in un sistema peer-to-peer, i servizi possono essere forniti da diversi nodi distribuiti in tutta la rete. Per partecipare a un sistema peer-to-peer, un nodo deve prima associarsi alla rete di pari. Una volta che un nodo è connesso alla rete, può iniziare a fornire servizi e richiedere servizi da altri nodi nella rete.



Sistema P2P

Cloud Computing

- Il cloud computing è un'architettura d'elaborazione che offre servizi di rete come memorizzazione di dati e applicazioni. Ad esempio, alcuni fornitori di Cloud hanno migliaia di server, milioni di macchine virtuali e immense quantità di memoria secondaria disponibili per gli utenti di Internet. Gli utenti pagano tariffe in base ai servizi e alle risorse che utilizzano. Ci sono molti tipi del cloud computing, tra cui:
  - **Cloud privato** gestito da una società per uso proprio.
  - ***Cloud pubblico,*** un cloud disponibile via Internet per gli utenti che sottoscrivono un abbonamento per i servizi.
  - **Cloud ibrido** che include componenti di cloud pubblici e privati.
  - **Infrastructure as a Service (IaaS),** server o dispositivi di memorizzazione utilizzabili, ad esempio, per fare il backup dei dati di produzione, fruibili attraverso Internet.
  - ***Platform as a Service (PaaS),*** un ambiente software per sviluppare applicazioni e utilizzabile tramite Internet, ad esempio, web e database server.

  - ***Software as Service (SaaS)***, una o più applicazioni, come word processor e fogli di calcolo, disponibili via Internet.

- In molti tipi di infrastrutture cloud, sono presenti sistemi operativi noti come Unix, Linux e Windows.

Struttura e organizzazione software dei sistemi operativi

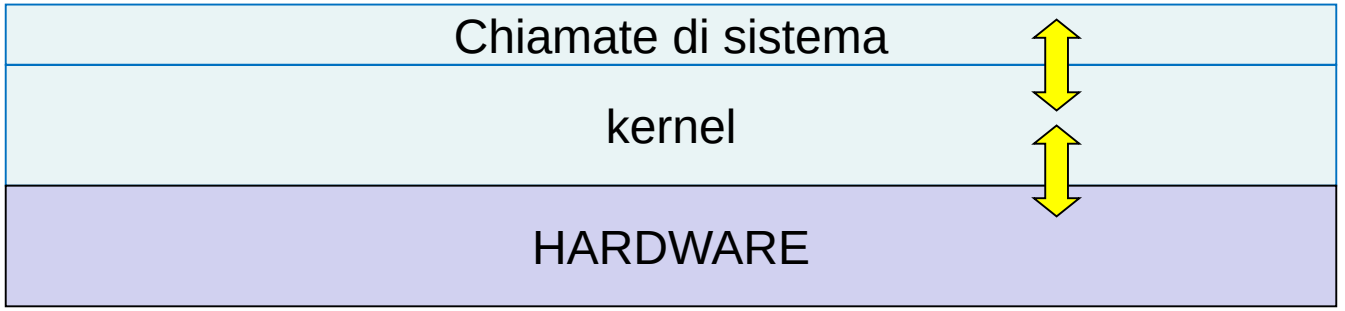
- Un sistema operativo deve svolgere molti compiti complessi. Per tale motivo dovrebbe essere progettato in modo tale che, oltre a funzionare correttamente, il suo codice sia facilmente modificabile.
- Ad alto livello, la progettazione del sistema sarà influenzata dalla scelta dell'hardware e dal tipo di sistema: batch, time sharing, real time, singolo utente, multiutente, e così via.
- Per la definizione e la progettazione di un sistema operativo si ricorre ai principi generali che sono stati sviluppati nel campo dell'ingegneria del software.
- Nella fase di progettazione è molto importante suddividere le operazioni che il sistema operativo deve svolgere in ***meccanismi*** *(tecniche)* e ***criteri*** *(politiche o strategie).* I meccanismi stabiliscono in che modo deve essere eseguito qualche compito; i criteri, invece, determinano in che modo utilizzare i meccanismi.

- Ad esempio, nei sistemi multiprogrammati il sistema operativo, per commutare la CPU da un processo all'altro, esegue un insieme di operazioni detto *cambio di contesto*, che comprende il salvataggio dei registri della CPU del processo che lascia la CPU ed il caricamento dei registri del nuovo processo che andrà in esecuzione
- Le operazioni di schedulazione della CPU, stabiliscono i criteri con cui assegnare la CPU ad un nuovo processo. Ad esempio la schedulazione potrebbe basarsi su una politica FIFO, su una politica basata sulle priorità che i processi possiedono, o su altri criteri.
- I meccanismi dovrebbero essere progettati in modo tale che siano separati dai criteri. Questo consente, nel caso si cambino i criteri, di mantenere ancora validi i meccanismi.
- Se ad esempio si decidesse di modificare la politica di schedulazione da FIFO ad una politica più complessa, sarebbe ancora possibile utilizzare i meccanismi già esistenti, senza modificarli.

- Una volta progettato il sistema operativo si deve realizzare.
- La scrittura di un sistema operativo dipende fortemente dall'architettura del hardware ed in particolare dal processore o processori utilizzati nel calcolatore.
- Come già descritto, molti processori sono dotati di istruzioni che possono essere eseguite in *modalità privilegiata o in modalità utente*. Questa caratteristica consente di realizzare ed organizzare il software in modo tale che solo il codice del sistema operativo possa eseguire le istruzioni privilegiate, proteggendo, pertanto, le componenti del sistema operativo stesso da un uso improprio o errato da parte dei programmi applicativi.
- Generalmente, i sistemi operativi, si scrivono con linguaggi di alto livello, come ad esempio il C ed il C++, con ristrette parti in linguaggio assembly, per poter accedere ai registri dei dispositivi hardware e realizzare funzioni compatte e veloci.
- I kernel di Linux e Windows sono scritti prevalentemente in C, anche se ci sono alcune piccole parti di codice assembly nello scheduler e nei driver di dispositivi.

Struttura monolitica

- Una semplice organizzazione del software, detta ***struttura monolitica***, consiste nel realizzare un insieme di funzioni ciascuna delle quali implementa un determinato servizio, attivabile tramite una o più chiamate di sistema. Spesso queste funzioni si scrivono in linguaggio assembly, per poter avere la massima velocità di esecuzione e una minore dimensione in termini di occupazione di memoria RAM.

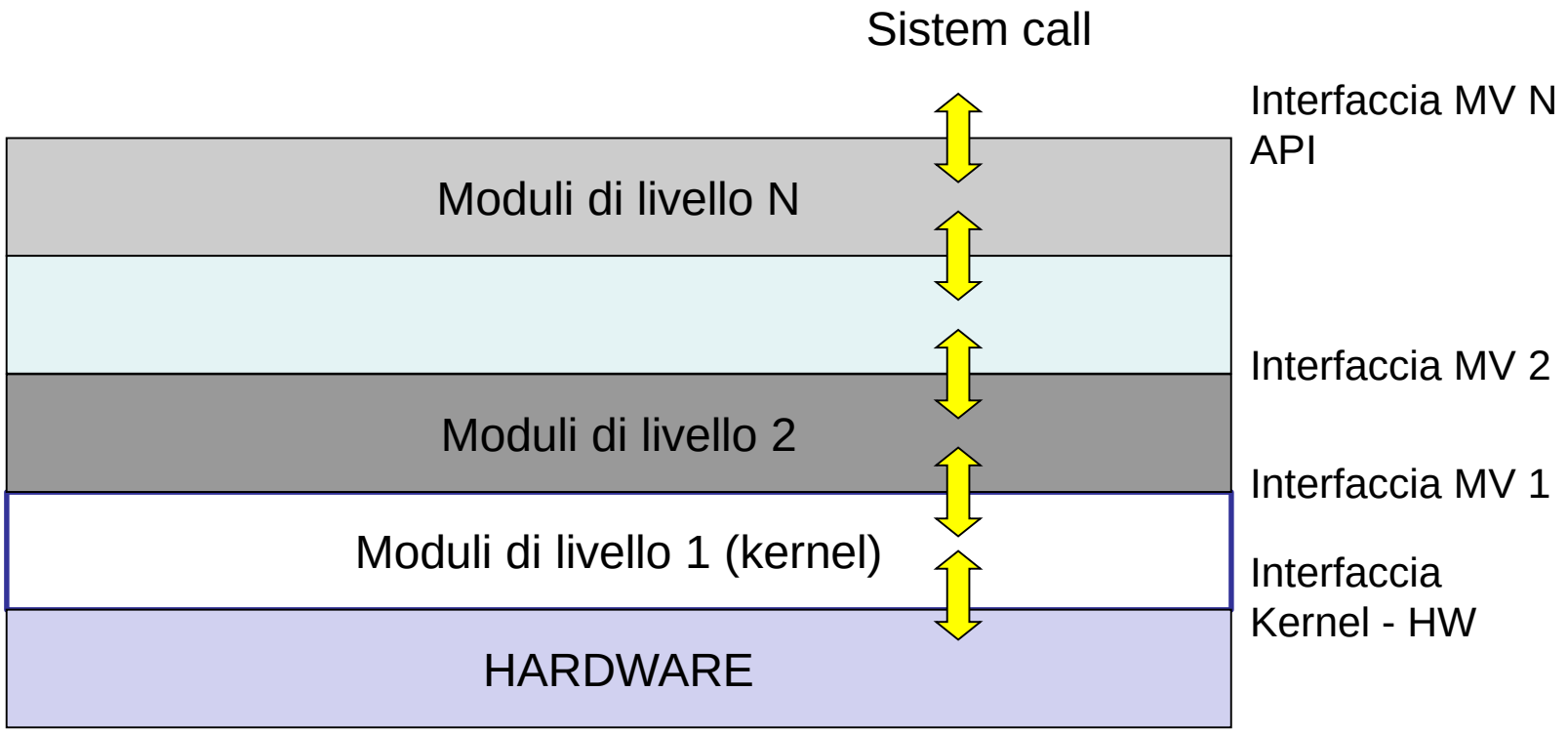


- Questa struttura, piuttosto semplice è stata usata nel sistema operativo Microsoft MS-DOS, un sistema operativo monoutente e mono tasking, scritto per microprocessori Intel 8088, 8086 e 80286, privi di modalità kernel.

- In assenza di modalità privilegiata, il programmatore può accedere a qualsiasi istruzione del microprocessore e quindi eseguire qualsiasi operazione come, ad esempio, scrivere dati in qualsiasi locazione di memoria, anche se riservata al sistema operativo. E' evidente che, in questi sistemi, un semplice errore di programmazione in un'applicazione può portare al crash del sistema.

Struttura stratificata

- Per la progettazione e lo sviluppo di sistemi più complessi si può ricorrere ai modelli e alle tecniche della programmazione strutturata o meglio ancora alla programmazione ad oggetti. I progettisti organizzano il sistema in un insieme di moduli, strutturandoli in vari livelli.
- Ciascun modulo di un livello utilizza le funzionalità offerte dai moduli di livello sottostante e fornisce a sua volta servizi ai moduli del livello superiore. Nei sistemi stratificati con il termine *kernel* si indica il livello che è a stretto contatto con l'hardware.



Struttura a livelli

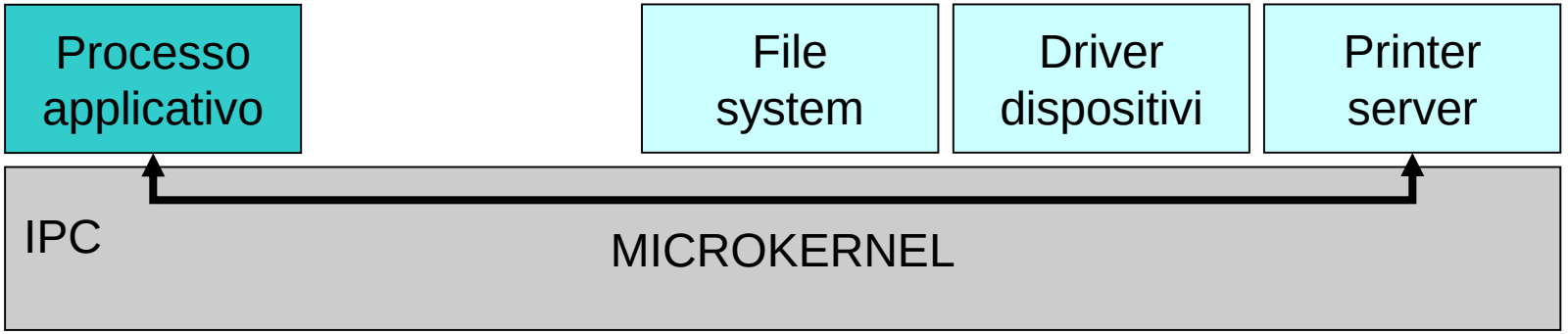
- La tecnica della stratificazione semplifica la fase di progettazione e rende più agevole apportare modifiche e correzioni al codice. Ogni strato può essere modificato, senza apportare cambiamenti ai restanti strati.
- Tuttavia, è richiesta un attenta e complessa analisi per stabilire quanti strati realizzare e scegliere quale funzionalità implementare in ciascun di essi.
- Inoltre, la stratificazione porta ad un funzionamento meno efficiente in termini di velocità di esecuzione e di occupazione di memoria. Ad esempio per eseguire un operazione, un programma applicativo, potrebbe effettuare una chiamata di sistema al livello sottostante, la quale, a sua volta, ne richiama un'altra, e questa un'altra ancora, e così. In altre parole, il programma applicativo per ottenere un servizio potrebbe attendere l'esecuzione di N funzioni di sistema.

- Bisogna tenere anche presente che nel passaggio da uno strato all'altro sono allocate strutture dati e parametri, con conseguente maggiore impegno di memoria. Per tale motivo, attualmente, si progettano sistemi stratificati con un limitato numero di strati.

Struttura a microkernel

- Come già descritto, per proteggere le componenti del SO è necessario che il processore sia dotato di istruzioni eseguibili in stato privilegiato per garantire che solo il codice del SO possa girare in stato privilegiato.
- D'altra parte, il fatto che solo il SO possa eseguire istruzioni privilegiate, rende il sistema più difficile da modificare.
- Per rendere più semplice, in particolare, il cambiamento delle politiche di gestione delle risorse è stata pensata la struttura a *microkernel*.

- Per gestire una risorsa sono definite due tipi di componenti del SO: le tecniche (meccanismi) per consentire la gestione della risorsa e le strategie di gestione (politiche), realizzate utilizzando i precedenti meccanismi.
- Nei sistemi a struttura a microkernel l'insieme dei meccanismi costituisce il microkernel, che è l'unico componente a girare nello stato privilegiato. Tutte le strategie sono implementate in programmi applicativi che girano nella modalità utente (non privilegiata). In tal modo questi programmi sono più facilmente modificabili ed espandibili. Quando un processo applicativo richiede una risorsa, interagisce con il relativo processo server mediante un insieme di chiamate di sistema detto ***IPC (Inter Process Comunication)***, fornite dal microkernel, che consentono la comunicazione tra processi.



La struttura a microkernel produce una perdita di efficienza, poiché per ogni operazione di comunicazione tra processi è necessario l’uso di chiamate di sistema.

Struttura modulare

- Attualmente, probabilmente la migliore tecnica per progettare e realizzare i sistemi operativi complessi si basa sulla programmazione orientata agli eventi e agli oggetti.
- Con tale tecnica si sviluppa il sistema in moduli, ciascuno dei quali svolge un particolare compito.
- Ogni modulo è implementato da un insieme di funzioni descritte da un'*interfaccia* che descrive le funzionalità svolte dal modulo e da un *corpo* che consiste nel codice che implementa le funzioni descritte nell'interfaccia.

Interfaccia Definizione di un insieme di funzioni implementate nel modulo
Corpo Implementazione delle funzionalità (nascoste all'esterno del modulo)

- Il codice del corpo di un modulo è nascosto al resto del sistema e si comunica con esso solo attraverso le funzioni della propria interfaccia. Il kernel si realizza in base ad un numero di componenti fondamentali, ai quali, se richiesto, se ne aggiungono altri dinamicamente durante la procedura di avvio o durante l’esecuzione. La tecnica di caricare dinamicamente moduli è attualmente usata dai sistemi operativi come ad esempio Windows, Linux, Solaris e Max OS X.

Struttura ibrida

- In realtà, pochi sistemi operativi adottano una struttura unica. Invece, generalmente, l'architettura è ibrida, costituita da una combinazione di diverse strutture, in modo da risolvere i problemi di prestazioni, sicurezza e usabilità.
- Ad esempio, Linux e altri sistemi Unix sono monolitici, perché avere il kernel in un unico spazio di indirizzamento offre prestazioni molto efficienti. Tuttavia, sono anche modulari, poiché nuove funzionalità possono essere aggiunte dinamicamente al kernel.
- Windows è in gran parte monolitico, soprattutto per motivi di prestazioni, ma conserva alcuni comportamenti tipici dei sistemi a microkernel, compreso il supporto per sottosistemi come win32 e POSIX, che sono eseguiti come processi in modalità utente. I sistemi Windows hanno anche il supporto per i moduli del kernel caricabili dinamicamente.