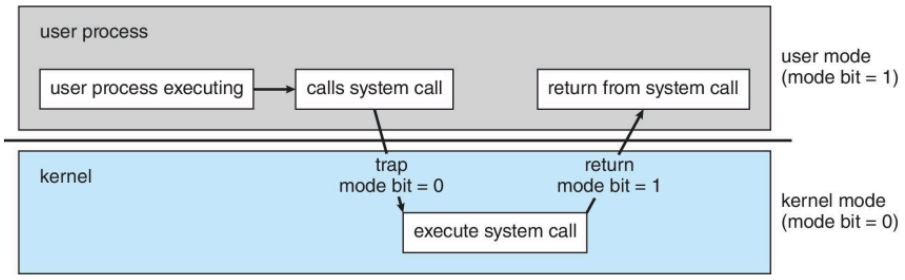
Lezione 3 (19-02-25)

Dual-Mode Operation

Un SO deve assicurarsi che un programma errato o dannoso non possa causare il malfunzionamento di altri programmi o del sistema operativo stesso. Per questo motivo è necessario distinguere tra l'esecuzione del codice del sistema operativo e l'esecuzione del codice utente. A tal fine, l'hardware fornisce un meccanismo che consente di differenziare tra le varie modalità di esecuzione. Sono necessarie almeno due modalità:

- **modalità utente**;
- **modalità di sistema**, detta anche *modalità kernel* o *modalità privilegiata*;
  Per indicare quale sia la modalità attiva, la CPU è dotata di un bit, detto *bit di modalità*, che assume il valore 0 in modalità kernel e 1 in modalità utente. Questo bit permette di stabilire se l'istruzione corrente viene eseguita per conto del sistema operativo o di un utente.
  Quando il sistema è in modalità utente e un'applicazione richiede un servizio al SO (tramite una *chiamata di sistema*), l'hardware deve commutare la CPU in modalità kernel.
  La **chiamata di sistema** avviene tramite una trap, un meccanismo che inserisce le istruzioni da eseguire in un registro della CPU, permettendo al kernel di sapere quale operazione deve essere svolta.



  All'avvio del sistema, il bit è impostato sulla modalità di sistema per consentire il caricamento del SO. Successivamente, il sistema passa in modalità utente per eseguire le applicazioni. Ogni volta che si verifica un interruzione o un'eccezione, l'hardware cambia la modalità da utente a sistema. Prima di restituire il controllo al programma utente, il SO ripristina la modalità utente, ponendo il bit a 1.
  Il *dual mode* consente la protezione del sistema e degli utenti dagli errori di altri utenti, definendo come *istruzioni privilegiate* quelle istruzioni di macchina che possono potenzialmente danneggiare il sistema.

Timer

Per garantire che il SO mantenga il controllo della CPU e per impedire che un programma utente entri in un ciclo infinito o non richieda servizi del sistema senza più restituire il controllo al sistema operativo si utilizza un **timer**.
Il timer è un dispositivo programmabile che genera un'interruzione a intervalli di tempo specificati. Esso può essere configurato per intervalli fissi o variabili. Un *timer variabile* funziona mediante un clock a frequenza fissa e un contatore: il SO assegna un valore iniziale al contatore, che viene decrementato a ogni impulso di clock; quando il valore arriva a 0, viene generata un'interruzione.
Prima di restituire il controllo al programma utente, il SO imposta un valore al timer. Se il tempo

assegnato scade, il SO genera un'interruzione, riprendendo il controllo della CPU.
Le istruzioni per modificare il valore del timer sono eseguibili solo in modalità privilegiata.

Process Management

Per eseguire le proprie operazioni, un processo necessita di risorse, come tempo di CPU, memoria, file e dispositivi di I/O. Queste risorse possono essere assegnate alla creazione del processo o durante la sua esecuzione.
Un programma è un'*entità passiva*, ovvero un file memorizzato su disco, mentre un processo è un'*entità attiva*. Un processo a singolo *thread* ha un contatore di programma (*program counter*) che indica la successiva istruzione da eseguire.
Un processo multi-thread possiede più contatori di programma, ognuno per un thread.
Un sistema è costituito da un insieme di processi, alcuni dei quali eseguono codice di sistema, altri codice utente. I processi possono essere eseguiti in modo concorrente, alternandosi sulla CPU o in parallelo su più unità di elaborazione.
Il sistema operativo è responsabile di alcune attività connesse alla gestione dei processi:

- creazione e cancellazione dei processi;
- schedulazione di processi e thread sulla CPU;
- sospensione e ripristino dei processi;
- sincronizzazione e comunicazione tra processi;
- gestione dei deadlock.

Memory Management

La memoria centrale è essenziale per il funzionamento del sistema. È un vasto vettore di parole, ognuna con un proprio indirizzo. È un archivio ad accesso rapido condiviso dalla CPU e dai dispositivi di I/O.
Per eseguire un programma, questo deve essere caricato in memoria e associato a indirizzi assoluti.
Il SO gestisce:

- monitoraggio delle aree di memoria in uso;
- allocazione e deallocazione della memoria;
- trasferimento di processi e dati tra memoria centrale e memoria di massa.

File-System Management

Il SO fornisce un'interfaccia uniforme per la gestione dei file, astraendo dalle caratteristiche fisiche dei dispositivi di memorizzazione. Un file è una raccolta di informazioni correlate e può essere organizzato in directory.
Il SO gestisce:

- creazione e cancellazione di file e directory;
- gestione dell'accesso ai file;
- associazione dei file ai dispositivi di memoria secondaria;
- creazione di copie di riserva (backup) dei file su dispositivi di memorizzazione non volatili.
  Alcuni SO, come Linux, permettono file-system a livello utente, senza interagire direttamente con l'hardware.

Mass-Storage Management

I sistemi utilizzano dischi magnetici e SSD per memorizzare programmi e dati.
Il SO si occupa di:

- montaggio (*mounting*) e smontaggio (*unmounting*) delle unità di memoria,
  smontare logicamente il disco prima di farlo fisicamente è importante per avvertire il SO di sincronizzare i dati, in modo da non rischiare la perdita di essi;
- gestione dello spazio libero;
- assegnazione dello spazio;
- scheduling del disco,
  (es. ho bisogno di un dato, ma devo aspettare che il disco finisca di fare il "giro" per leggere il dato con la testina, allora per ottimizzare i tempi leggo gli altri dati di cui ho bisogno);
- partizionamento;
- protezione.

Caching

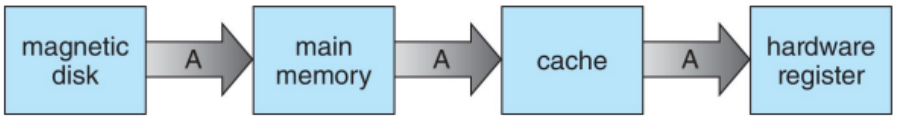
Le informazioni sono mantenute in un sistema di memoria come la memoria centrale; al momento del loro uso si copiano temporaneamente in un'unità più veloce, ovvero la **cache**. Prima di accedere ad una determinata informazione si controlla se è già presente all'interno della cache: in questo caso si utilizza direttamente la copia contenuta nella cache, altrimenti la si preleva dalla memoria centrale e la si copia nella cache, poiché si suppone che questa informazione presto servirà ancora. Si copiano anche i dati vicini perché è probabile che serviranno anche quelli. Quando qualcosa viene inserito nella cache, deve essere rimosso qualcos'altro; esistono degli algoritmi per capire quali dati devono entrare e quali uscire. Inoltre, i registri programmabili all'interno della CPU rappresentano per la memoria centrale una cache ad alta velocità. Vengono implementati degli algoritmi di assegnazione e aggiornamento dei registri in modo da stabilire quali informazioni mantenere nei registri e quali nella memoria centrale. Esistono cache interamente gestite dall'hardware del sistema, in quanto senza la CPU dovrebbe attendere parecchi cicli prima che un'istruzione sia prelevata dalla memoria. Per gli stessi motivi, la maggior parte dei sistemi è dotata di una o più cache di dati nella gerarchia delle memorie.

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

1. *registri*: memoria più veloce, ma la più piccola. Contengono pochi dati e vengono usati direttamente dal processore per eseguire operazioni velocemente. È gestita dal compilatore (software che trasforma il codice in istruzioni eseguibili);

2. *cache*: memoria intermedia tra i registri e la RAM. Serve per velocizzare l'accesso ai dati più usati. Gestita dall'hardware;
3. *memoria principale*: memoria che il computer utilizza per eseguire i programmi. (es. quando viene aperta un'app, i dati vengono caricati qui). Gestita dal SO.
4. *disco a stato solido*: tipo di memoria di archiviazione permanente. Più veloce di un disco rigido tradizionale, ma più costoso. Gestito dal SO.
5. *disco magnetico*: il classico hard disk, molto più capiente ma anche più lento rispetto a un SSD. Gestito dal SO.
   Il movimento delle informazioni tra i vari livelli della gerarchia può essere sia implicito sia esplicito, a seconda dell'hardware e del sistema operativo. Ad esempio, il trasferimento dei dati tra la cache e i registri della CPU è di solito svolto dall'hardware del sistema senza alcun intervento del sistema operativo, mentre il trasferimento dei dati dai dischi alla memoria è di solito gestito dal SO.
   In una struttura gerarchica può accadere che gli stessi dati siano mantenuti contemporaneamente in diversi livelli del sistema di memorizzazione.
   Esempio:

   > Si supponga di dover modificare il valore di una variabile contenuta in un file su un disco magnetico. L'operazione di modifica prevede innanzitutto l'esecuzione di un'operazione di I/O per copiare nella memoria centrale il blocco di disco contenente il valore della variabile. In seguito, si copia il valore all'interno della cache e poi in uno dei registri interni della CPU. Quindi, esistono diverse copie della variabile memorizzate nei vari dispositivi. Dopo la modifica del valore contenuto nel registro interno, il valore della variabile sarà diverso da quello assunto dalle altre copie della stessa variabile. Le diverse copie avranno lo stesso valore solamente dopo che il nuovo valore sarà stato riportato dal registro interno della CPU alla copia della variabile residente nel disco.



In un sistema elaborativo che ammette l'esecuzione di un solo processo alla volta non ci sono difficoltà. Nei sistemi *multitasking* si deve prestare più attenzione in quanto bisogna assicurare che ogni processo che desideri accedere alla variabile ottenga dal sistema il valore aggiornato più di recente.
Negli ambienti *multiprocessore* possono esistere più copie simultanee della variabile mantenute in cache differenti (ciascuna CPU ha la propria cache). Dal momento che le diverse unità di elaborazione possono operare in parallelo, è necessario che l'aggiornamento del valore di una qualsiasi delle copie di A contenute nelle cache locali si rifletta immediatamente in tutte le cache in cui risiede la variabile. Questa situazione è nota come **coerenza della cache**, solitamente si risolve a livello dell'hardware del sistema.
In un sistema distribuito può accadere che più copie (dette *repliche*) dello stesso file siano mantenute in differenti calcolatori, dislocati in luoghi fisici diversi. Essendoci la possibilità di accedere e modificare in modo concorrente queste repliche, è necessario fare in modo che ogni modifica a una qualunque replica si rifletta il prima possibile sulle altre.

I/O Subsystem

Il SO astrae le caratteristiche dei dispositivi di I/O mediante:

- gestione della memoria per buffer, cache e *spooling*;
- interfaccia generica per i driver;
- driver specifici per i dispositivi.

Spooling := (*Simultaneous Peripheral Operations On-Line*) è una tecnica in cui i dati vengono temporaneamente memorizzati in una coda su disco prima di essere inviati al dispositivo di destinazione. Ad esempio, in una stampante, i lavori di stampa vengono accodati su disco e stampati in ordine senza bloccare il sistema.
