Sistemi operativi

Giacomo Fantoni
Telegram: @GiacomoFantoni
Github: https://github.com/giacThePhantom/SistemiOperativi

23 febbraio 2020

Indice

Capitolo 1

Introduzione

Un sistema operativo è un insieme di programmi che agiscono come intermediario tra l'hardware e l'uomo per facilitare l'uso del computer, rendere efficiente l'uso dell'hardware e evitare conflitti nell'allocazione di risorse tra hardware e software. Offre pertanto un ambiente per controllare e coordinare l'utilizzo dell'hardware da parte dei programmi applicativi. I suoi compiti principali sono di gestore di risorse e di controllare l'esecuzione dei programmi e il corretto utilizzo del sistema. Nel

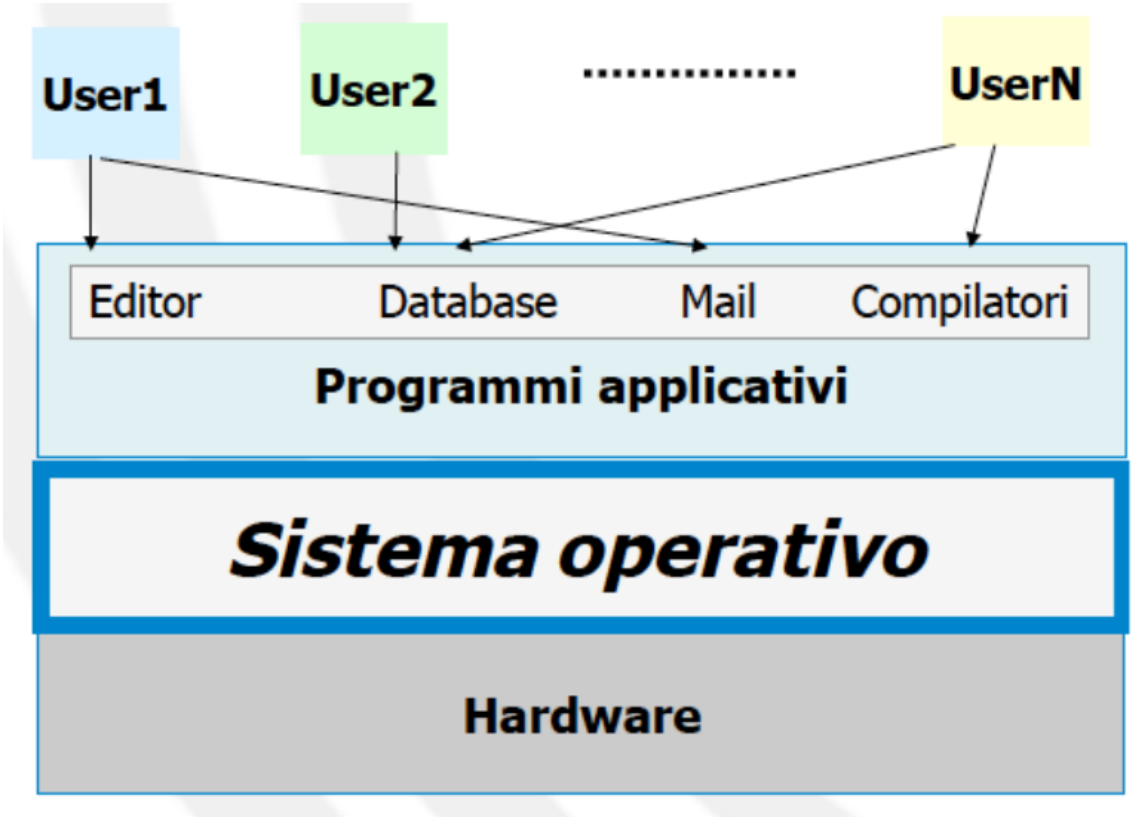

Figura 1.1: Stack del sistema operativo

progettare un sistema operativo si deve tipicamente fare un trade-off tra l'astrazione che semplifica l'utilizzo del sistema e l'efficienza.

1.1 Storia dei sistemi operativi

Si possono identificare 5 generazioni di calcolatori che riflettono direttamente l'evoluzione dei sistemi operativi dovuta all'aumento dell'utilizzo del processore.

1.1.1 Prima generazione (1946-1955)

In questa generazione i calcolatori erano enormi e a valvole, non esisteva il sistema operativo e l'operatore del calcolatore era equivalente al programmatore. L'accesso alla macchina era gestito tramite prenotazioni e i programmi venivano eseguiti da console caricando in memoria un'istruzione alla volta agendo su interruttori. Il controllo degli errori era fatto attraverso spie della console. Il processing era seriale.

Evoluzione

Durante la prima generazione si diffondono periferiche come il lettore/perforatore di schede, nastri e stampanti che rendono necessari programmi di interazione con periferiche detti device driver. Viene sviluppato del software come librerie di funzioni comuni e compilatori, linker e loader. Queste evoluzioni portano a una scarsa efficienza in quanto pur essendo la programmazione facilitata le operazioni erano complesse con tempi di setup elevati e un basso utilizzo relativo della CPU per eseguire il programma.

1.1.2 Seconda generazione (1955-1965)

In questa generazione si introducono i transistor nei calcolatori. Viene separato il ruolo di programmatore e operatore eliminando lo schema a prenotazione e il secondo permette di eliminare dei tempi morti. I programmi o jobs simili nell'esecuzione vengono raggruppati in batch in modo da aumentare l'efficienza ma aumentando i problemi in caso di errori o malfunzionamenti.

Evoluzione

Nasce l'automatic job sequencing in cui il sistema si occupa di passare da un job all'altro: il sistema operativo fa il lavoro dell'operatore e rimuove i tempi morti. Nasce pertanto il monitor residente, il primo esempio di sistema operativo, perennemente caricato in memoria. Le componenti del monitor erano i driver per i dispositivi di I/O, il sequenzializzatore dei job e l'interprete delle schede di controllo (per la loro lettura ed esecuzione). La sequenzializzazione avviene tramite un linguaggio di controllo o job control language attraverso schede o record di controllo.

Limitazioni

L'utilizzo del sistema risulta ancora basso a causa del divario di velocità tra I/O e CPU. Una soluzione è la sovrapposizione delle operazioni di I/O e di elaborazione. Nasce così l'elaborazione off-line grazie alla diffusione di nastri magnetici capienti e veloci. La sovrapposizione avviene su macchine diverse: da scheda a nastro su una macchina e da nastro a CPU su un'altra. La CPU viene limitata ora dalla velocità dei nastri, maggiore di quella delle schede.

Sovrapposizione tra CPU e I/O

È possibile attraverso un opportuno supporto strutturale far risiedere sulla macchina le operazioni off-line di I/O e CPU.

Polling Il polling è il meccanismo tradizionale di interazione tra CPU e I/O: avviene l'interrogazione continua del dispositivo tramite esplicite istruzioni bloccanti. Per sovrapporre CPU e I/O è necessario un meccanismo asincrono o richiesta I/O non bloccante come le interruzioni o interrupt e il DMA (direct memory access).

Interrupt e I/O In questo caso la CPU programma il dispositivo e contemporaneamente il dispositivo controllore esegue. La CPU, se possibile prosegue l'elaborazione. Il dispositivo segnala la fine dell'elaborazione alla CPU. La CPU riceve un segnale di interrupt esplicito e interrompe l'istruzione corrente salvando lo stato, salta a una locazione predefinita serve l'interruzione trasferendo i dati e riprende l'istruzione interrotta.

DMA e I/O Nel caso di dispositivi veloci gli interrupt sono molto frequenti e porterebbero a inefficienza. Si rende pertanto necessario creare uno specifico controllore hardware detto DMA controller che si occupa del trasferimento di blocchi di dati tra I/O e memoria senza interessare la CPU. Avviene pertanto un solo interrupt per blocco di dati.

Buffering Si dice buffering la sovrapposizione di CPU e I/O dello stesso job. Il dispositivo di I/O legge o scrive più dati di quanti richiesti e risulta utile quando la velocità dell'I/O e della CPU sono simili. Nella realtà i dispositivi di I/O sono più lenti della CPU e pertanto il miglioramento è marginale.

Spooling Si dice spooling (simultaneous peripheral operations on-line))la sovrapposizione di CPU e I/O di job diversi, Nasce un problema in quanto i nastri magnetici sono sequenziali e pertanto il lettore di schede non può scrivere su un'estremità del nastro mentre la CPU legge dall'altra. Si devono pertanto introdurre dischi magnetici ad accesso causale. Il disco viene utilizzato come un buffer unico per tutti i job. Nasce il paradigma moderno di programma su disco che viene caricato in memoria, la pool di job e il concetto di job scheduling (la decisione di chi deve o può essere caricato su disco).

1.1.3 Terza generazione (1965-1980)

In questa generazione viene introdotta la multiprogrammazione e i circuiti integrati. La prima nasce dal fatto che un singolo job è incapace di tener sufficientemente occupata la CPU e pertanto si rende necessaria una loro competizione. Sono presenti più job in memoria e le fasi di attesa vengono sfruttate per l'esecuzione di un nuovo job. Con la presenza di più job nel sistema diventa possibile modificare la natura dei sistemi operativi: si passa ad una tendenza a sodisfare molti utenti che operano interattivamente e diventa importante il tempo di risposta di un job (quanto ci vuole perchè inizi la sua esecuzione). Nasce pertanto il multitasking o time sharing, estensione logica della multiprogrammazione in cui l'utente ha l'impressione di avere la macchina solo per sè e si migliora l'interattività con la gestione di errori e l'analisi di risultati. Nascono i sistemi moderni con tastiera che permette decisioni dell'evoluzione del sistema in base ai comandi dell'utente e un monitor che permette un output immediato durante l'esecuzione. Il file system inoltre è un'astrazione del sistema operativo per accedere a dati e programmi.

Protezione

Con la condivisione si rende necessario introdurre delle capacità di protezione per il sistema:

- I/O: programmi diversi non devono usare il dispositivo contemporaneamente, viene realizzata tramite il modo duale di esecuzione: modo user in cui i job non possono accedere direttamente alle risorse di I/O e modo supervisor o kernel in cui il sistema operativo può accedere a tali risorse. Tutte le operazioni di I/O sono privilegiate: le istruzioni di accesso invocano una system call, un interrupt software che cambia la modalità da user a supervisor e al termine della system call viene ripristinata la modalità supervisor.
- Memoria: un programma non può leggere o scrivere ad una zona di memoria che non gli appartiene: realizzata associando dei registri limite ad ogni processo, che possono essere modificati unicamente dal sistema operativo con istruzioni privilegiate.
- CPU: prima o poi il controllo della CPU deve tornare al sistema operativo, realizzata attraverso un timer legato ad un job, al termine del quale il controllo passa al monitor.

1.1.4 Quarta generazione (1980-1990)

- Diffusione di sistemi operativi per PC e workstation, utilizzo personale degli elaboratori e nascita delle interfacce grafiche (GUI).
- Sistemi operativi di rete in cui esiste una separazione logica delle risorse remote in cui l'accesso alle risorse remote è diverso rispetto a quello delle risorse locali.
- Sistemi operativi distribuiti: le risorse remote non sono separate logicamente e l'accesso alle risorse remote e locali è uguale.

Quinta generazione (1990- oggi

Sistemi real-time vincolati sui tempi di risposta del sistema, sistemi operativi embedded per applicazioni specifiche, per piattaforme mobili e per l'internet of things.

Capitolo 2

Componenti di un sistema operativo

Le componenti di un sistema operativo si identificano in base alla loro funzione.

Gestione dei processi

Si intende per processo un programma in esecuzione che necessita di risorse e viene eseguito in modo sequenziale un'istruzione alla volta. Si differenzia tra processi del sistema operativo e quelli utente. Il sistema operativo è responsabile della creazione, distruzione, sospensione, riesumazione e ella fornitura di meccanismi per la sincronizzazione e la comunicazione tra processi.

Gestione della memoria primaria

La memoria primaria conserva dati condivisi dalla CPU e dai dispositivi di I/O. Un programma deve essere caricato in memoria prima di poter essere eseguito. Il sistema operativo è responsabile della gestione dello spazio di memoria (quali parti e da chi sono usate), della decisione su quale processo caricare in memoria quando esiste spazio disponibile e dell'allocazione e rilascio dello spazio di memoria.

Gestione della memoria secondaria

Essendo la memoria primaria volatile e piccola si rende necessaria una memoria secondaria per mantenere grandi quantità di dati in modo permanete. È formata tipicamente da un insieme di dischi magnetici. Il sistema operativo è responsabile della gestione dello spazio libero su disco, dell'allocazione dello spazio su disco e dello scheduling degli accessi su disco.

Gestionde dell'I/O

Il sistema operativo nasconde all'utente le specifiche caratteristiche dei dispositivi di I/O, il cui sistema consiste di un sistema per accumulare gli accessi ai dispositivi (buffering), una generica interfaccia verso i device driver, con device driver specifici per alcuni dispositivi.

Gestione dei file

Le informazioni sono memorizzate su supporti fisici diversi controllati da driver con caratteristiche diverse. Si crea pertanto un file, un'astrazione logica per rendere conveniente l'uso di memoria non volate grazie alla raccolta di informazioni correlate. Il sistema operativo è responsabile della creazione e cancellazione di file e directory, supporto di operazioni primitive per la gestione di file e directory, corrispondenza tra file e spazio fisico su disco, salvataggio delle informazioni a scopo di backup.

Protezione

Si intende con protezione un meccanismo per controllare l'accesso alle risorse da parte di utenti e processi. Il sistema operativo è responsabile della definizione di accessi autorizzati e non, definizione dei controlli da imporre, fornitura di strumenti per verificare le politiche di accesso.

Rete (sistemi distribuiti)

Si intende per sistema distribuito una collezione di elementi di calcolo che non condividono nè la memoria nè un clock: le risorse di calcolo vengono connesse tramite una rete. Il sistema operativo è responsabile della gestione in rete delle varie componenti.

Interprete dei comandi (Shell)

La maggior parte dei comandi vengono forniti dall'utente al sistema operativo tramite istruzioni di controllo che permettono di creare e gestire processi, I/O, disco, memoria, file system, protezioni e rete. Il programma che legge e interpreta questi comandi è l'interprete dei comandi che legge ed esegue la successiva istruzione di controllo o comando.

2.1 Interprete

L'interprete principalmente ha il compito di prendere comandi specificati dall'utente e eseguirli. Il codice dei comandi è specificato dall'interprete stesso ed è costituito da programmi predefiniti a cui si riferisce attraverso il loro nome.

2.1.1 Interfaccia utente

L'interprete può fornire diversi tipi di interfaccia utente: command-line (CLI), Graphics User Interface (GUI) o Batch. CLI permette di digitare direttamente comandi di testo come avveniva in UNIX. È spesso implementata nel kernel e alcune volte come programma di sistema con alcune varianti dette shells come c o bourne. Il desktop è una metafora di interfaccia utente, generalmente legata a mouse, tastiera e schermo con le icone che rappresentano file, programmi e azioni. I vari tasti del mouse puntati sull'oggetto visualizzato nell'interfaccia possono eseguire azioni diverse.

2.2 System call

Le system calls forniscono l'interfaccia tra i processi e i servizi offerti dal sistema operativo. Sono tipicamente scritte in linguaggio di alto livello come C o C++, qualcuna in assembler,

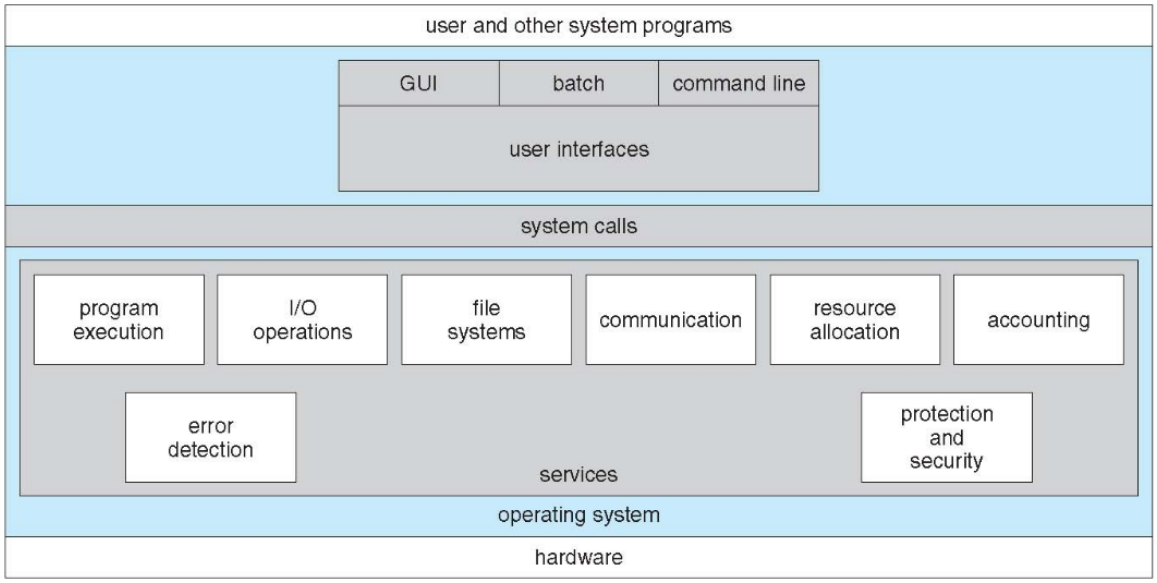

Figura 2.1: Stack del sistema operativo

2.2.1 Implementazione

Tipicamente un numero viene associato ad ogni system call e l'interfaccia delle chiamate di sistema mantiene una tabella indicizzata secondo essi. L'interfaccia invoca la system call usata nel kernel del sistema operativo e poi ritorna lo stato della system call e gli eventuali valori di ritorno. Il chiamante non ha necessità di conoscere come la system call è implementata. Mascherano pertanto i dettagli implementativi del sistema operativo fornendo un livello di astrazione intermedio. Sono chiamate da programmi attraverso l'interfaccia per la programmazione di applicazioni (Application Program Interface - API) di alto livello piuttosto che usate direttamente.

API

Le due APIs più comuni sono Win32 API per Windows e POSIX API per sistemi POSIX (portable operating-system interface for Unix) e le Java API per la Java virtual machine. Le API dovrebbero garantire la portabilità delle applicazioni almeno sullo stesso tipo di API.