Sistemi operativi

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

18 marzo 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 controllore dell'esecuzione dei programmi e il corretto utilizzo del sistema. La struttura dei sistemi operativi è soggetta a notevole variabilità ad è adattabile a criteri di organizzazione estremamente differenti. È pertanto un programma sempre in esecuzione sul calcolatore che generalmente viene chiamato kernel al quale si aggiungono programmi di sistema e programmi applicativi. Nel progettare un sistema operativo si deve tipicamente fare un trade-off tra l'astrazione che semplifica l'utilizzo del sistema e l'efficienza.

Componenti

Un sistema di calcolo si può dividere in 4 componenti:

- Dispositivi fisici: sono composti dall'unità centrale di elaborazione (CPU), dalla memoria e dall'I/O.
- Programmi applicativi: definiscono il modo in cui utilizzare le risorse per risolvere i problemi computazionali da parte degli utenti.
- Sistema operativo: Controlla e coordina l'udo dei dispositivi da parte degli utenti.
- Utenti.

1.0.1 Punti chiave nel progetto di calcolatori

Punto di vista dell'utente

La percezione di un calcolatore dipende dall'interfaccia impiegata. Il più comune metodo di utilizzo è il PC composto da schermo, tastiera e mouse. Il sistema operativo in questo caso si progetta considerando la facilità di utilizzo con qualche attenzione alle prestazioni ma non all'utilizzo delle risorse. Nel caso di un utente che utilizza terminali connessi ad un mainframe condividendo risorse con altri utenti il sistema operativo andrebbe ottimizzato per massimizzare l'utilizzo delle risore.

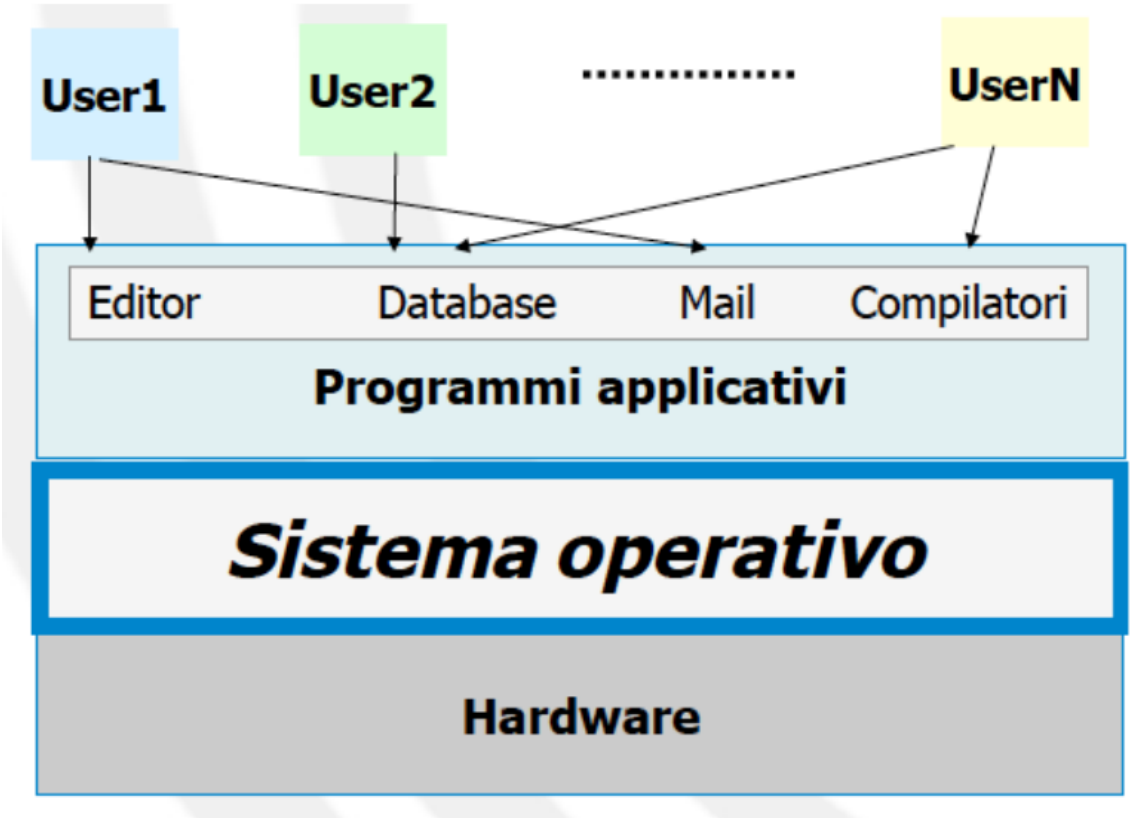

Figura 1.1: Stack del sistema operativo

Punto di vista del sistema

Il sistema operativo è il programma collegato più strettamente ai suoi elementi fisici ed è assimilabile ad un assegnatore di risorse o come programma di controllo che gestisce l'esecuzione dei programmi utente in modo da impedire che si verifichino errori o che il calcolatore sia utilizzato in modo scorretto.

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

Un sistema operativo offre un ambiente on cui eseguire i programmi e fornire servizi che naturalmente variano in base al sistema operativo. Si possono comunque identificare alcune classi di servizi comuni.

2.1 Servizi di gestione

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 e fornisce meccanismi per la sincronizzazione.

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), ovvero della decisione su quale processo caricare in memoria in base allo 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 permanente. È formata tipicamente da un insieme di dischi magnetici (che stanno per essere sostituiti dai dischi a stato solido -SSD- più veloci e performanti). 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 per motivi di efficienza e protezione. Viene impegato 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 volatile grazie alla raccolta di informazioni correlate. Il sistema operativo è responsabile della creazione e cancellazione di file e directory, del supporto di operazioni primitive per la loro gestione (copia, incolla, modifica), della corrispondenza tra file e spazio fisico su disco e del 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 deve definire quali sono gli accessi autorizzati e quali no, i controlli da imporre e fornire gli strumenti per verificare le politiche di accesso. La sicurezza di un sistema operativo comincia con l'obbligo di identificazione di ciascun utente che permette l'accesso alle risorse.

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.

2.2 Interprete dei comandi (shell)

Vi sono due modi fondamentali per gli utenti di comunicare con il sistema operativo: un primo basato su un'interfaccia a riga d comando (o interprete dei comandi) e un secondo bastato su un'interfaccia grafica o GUI. Il primo lascia inserire direttamente agli utenti le istruzioni che il sistema deve eseguire. L'interprete dei comandi è più comunemente conosciuto come shell. La funzione principale dell'interprete è quella di prelevare ed eseguire il successivo comando impartito dall'utente. A questo livello si utano nuovi comandi per la gestione dei file che possono essere implementati internamente all'interprete o attraverso programmi speciali. Nel secondo caso l'interprete non capisce il comando in sè ma prende il nome per caricare l'opportuno file in memoria per eseguirlo.

Interfaccia grafica

L'interfaccia grafica è una modalità di comunicazione tra utente e il sistema operativo. È più intuitiva della riga di comando e la GUI è l'equivalente del desktop e rimane strettamente legata a mouse, tastiera e schermo. Puntando le icone col mouse è possibile accedere a file, cartelle e applicazioni.

2.2.1 System calls

Le chiamate di sistema costituiscono l'interfaccia di comunicazione tra il processo e il sistema operativo. Sono tipicamente scritte in linguaggi di alto livello come *C* o *C++*. I programmatori non si devono preoccupare dei dettagli di implementazione delle *sys.call* in quanto solitamente utilizzano un'API (application program interface) che specifica un'insieme di funzioni a disposizione dei programmatori e dettaglia i parametri necessari all'invocazione di queste funzioni e i valori restituiti. Le due API più comuni sono *win32* e *POSIX API*, rispettivamente per Windows e UNIX. I parametri delle system calls possono essere pasati per valore o riferimento, ma vanno fisicamente messi da qualche parte: vengono pertanto posizionati nei registri (molto veloci, ma pochi e di dimensione fissa) nello stack del programma o in una tabella di memoria il cui indirizzo è passato in un registro o nello stack. Le system calls possono essere implementate in due modi:

- L'interprete legge il comando e cerca all'interno della shell per cercare il programma da avviare, non viene utilizzata in quanto rende necessario modifiche al kernel e non è efficiente.
- L'interprete legge il comando e possiede una tabella che collega tale comando al programma da avviare.

Le system calls si differiscono in cotrollo dei processi, gesione dei file, dei dispositivi, delle comunicazioni e della protezione. Un processo inoltre deve essere sia in grado di essere chiuso normalmente (*end*) che in modo anomalo (*abort*), con la conseguente generazione di un messaggio di errore e copia dello stato del processo abortito.

Capitolo 3

Architettura di un sistema operativo

Un principio importante è la separazione tra meccanismi e criteri o policy: i primi determinano come eseguire qualcosa, mentre i secondi cosa si deve fare. Questa distinzione è importante ai fini della flessibilità in quanto i criteri sono soggetti a cambiamenti di luogo e tempo.Principi importanti da tenere a mente durante lo sviluppo di sistemi operativi è il KISS (keep it small and simple), semplice dal punto di vista del codice, per mantenere affidabilità e mantenibilità e il POLA (principle of least privilege): un programm deve poter accedere unicamente ai dati strettamente necessari, fondamentale per il mantenimento di sicurezza e affidabilità.Questi cambiamenti devono richiedere il cambio di meccanismi solo nel caso pessimo. Le principali tipologie di architettura di sistemi operativi sono:

- Sistemi monolitici: sistemi senza gerarchia e con un unico strato software tra utente e hardware, tutti i componenti sono sullo stesso livello e possono chiamarsi vicendevolmente. In questa tipologia il codice dipende direttamente dall'architettura hardware e rende test e debugging complesso.
- Sistemi a struttura semplice: si ha un minimo di gerarchia e di struttura, non esiste ancora la suddivisione modalità utente e modalità kernel. Questa struttura è mirata a ridurre i costi di sviluppo e manutenzione.
- Sistema operativo a livelli. I servizi sono organizzati su livello gerarchici con al livello più alto l'interfaccia utente e al più basso l'hardware. Ogni livello può utilizzare funzioni di livelli inferiori. La modularità rende più semplice la manutenzione, ma diminuisce l'efficienza e richiede un'attenta definizione dei livelli.
- Sistemi basati su kernel: vengono utilizzati due livelli, i cui servizi sono distinti tra kernel e non-kernel. Presenta i vantaggi del sistema a livelli come modularità ma senza avere troppi livelli. Tra i servizi al di fuori del kernel non si trova nessuna organizzazione e si tende a pensare al kernel come a una struttura monolitica.
- Sistemi a micro-kernel: i micro-kernel sono un insieme di piccoli kernel che svolgono poche funzioni fondamentali. Occupano meno memoria e sono più affidabili e mantenibili, ma presentano scarse prestazioni: ogni volta che si deve accedere ad un programma applicativo si deve fare un cambio tra modalità kernel a modalità utente e viceversa una volta terminato il

processo. Vengono utilizzati da quando le prestazioni di CPU e memoria sono sufficienti a non far percepire all'utente il cambio di modalità. La modularità offre inoltre maggiore sicurezza e portabilità.

3.1 macchine virtuali

Le macchine virtuali sono introdotte nel 1972 da IBM come estremizzazione dell'approccio a livelli persata per offrire un sistema di timesharing multiplo ovvero che permette la multiprogrammazione e una macchina estesa che abbia un interfaccia più semplice del solo hardware. La base della macchina virtuale è la separazione di questi due aspetti. La sua parte centrale era il virtual machine monitor che permette la multiprogrammazione offrendo diverse macchine virtuali al livello superiore. Un tipo di macchina virtuale utilizza un type 1 hypervisor, usato comuniemente che si trova sull'hardware e permette di eseguire diversi sistemi operativi sulla stessa macchina. Il type 2 hypervisor viene utilizzato su un sistema operativo host nel quale l'hypervisor installa il sistema operativo guest in un disco virtuale che il sistema host vede come un file di grandi dimensioni. La differenza tra i due tipi di hypervisor sta nel fatto che quello di tipo 1 si trova direttamente sull'hardware, mentre il tipo 2 viene creato in un sistema operativo host.

3.2 Processi e thread

Attributi (Process Control Block): contiene un puntatore alla cella di memoria che contiene l'immagine, contiene lo stato del processo in un determinato momento, contiene i registri, le informazioni relative allo stato dell'I/O. Un processo può essere in diversi stati. All'inizio il processo viene creato, poi può essere in esecuzione se gli viene assegnata la CPU o non in esecuzione se non gli viene assegnata la CPU. Il Dispatcher assegna la CPU ai processi che sono pronti, ma non in esecuzione (posso avere diversi processi in memoria, ma solo uno per volta usa la CPU e quindi è effettivamente in esecuzione, a meno di CPU multicore). Quando un processo è pronto e viene creato viene messo nella ReadyQueue (oppure nella coda di un dispositivo cioè la coda in cui viene messo un processo che sta aspettando di accedere a un determinato dispositivo). In realtà esistono diverse code in cui può essere messo un processo. Dispatch e Scheduler sono due componenti diversi, lo scheduler sceglie mentre il dispatcher implementa questa cosa. Context-Switch: salvo tutto quello che c'era nel PCB, tutto quello che stavo utilizzando al momento dell'esecuzione. Due operazioni fondamentali che fa un sistema operativo è la creazione e la terminazione del processo. Ogni processo può creare altri processi e questi prendono il nome di processi figli. Il figlio può essere creato in modalità sincrona, cioè finché il figlio è in vita io non faccio niente, oppure in modalità asincrona cioè io creo il figlio e continuo la mia esecuzione. Con la fork il figlio lo creo esattamente uguale al padre, con la exec posso caricare sul figlio un programma diverso rispetto al padre. Con la wait creo un'esecuzione sincrona tra padre e figlio. Una fork può fallire perché o il padre non ha abbastanza memoria o perché non ha i privilegi per creare un figlio. L'exec cambia l'immagine in memoria del figlio.