Microbiologia generale

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

19 febbraio 2020

Indice

1 Introduzione **2**
1.1 Storia dei sistemi operativi 3
1.1.1 Prima generazione (1946-1955) 3
1.1.2 Seconda generazione (1955-1965) 3
1.1.3 Terza generazione (1965-1980) 4

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

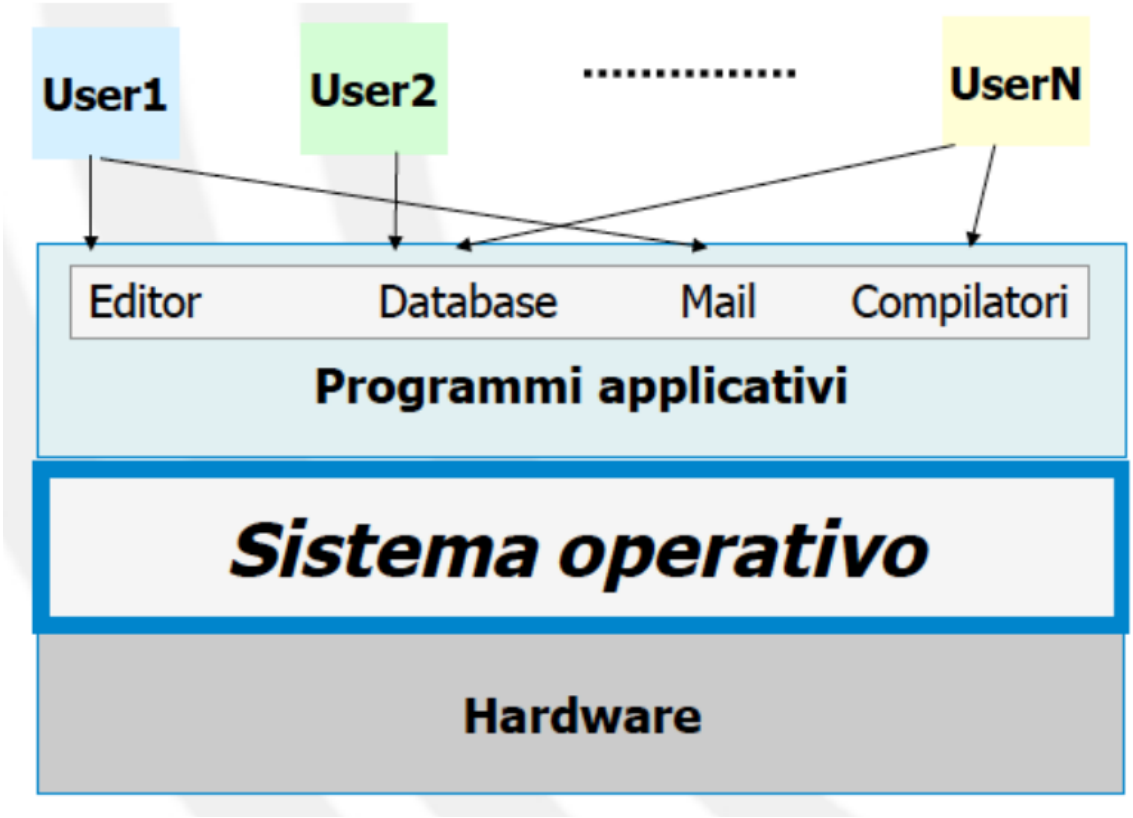

Figura 1.1: Stack del sistema operativo

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.