Modulo 2 - Architettura dei sistemi operativi

L1 - Funzioni di un sistema operativo

Un calcolatore, come abbiamo visto, è composto da CPU, memoria centrale e da un insieme di periferiche che consentono l'interazione con il mondo esterno: questo è l'aspetto hardware che il Sistema Operativo deve gestire.

Astrazione

L'innalzamento del livello di astrazione dei componenti permette di semplificare l'accesso alle risorse e mostrare ai programmi e agli utenti un comportamento più semplice delle varie risorse, nascondendo dietro l'astrazione i dettagli di gestione delle periferiche.

Questo consente di individuare i dati individuandone un nome in modo logico all'interno del nostro sistema e non più sapendo dove si trovano fisicamente le informazioni nelle risorse.

Virtualizzazione

Questo ha come obiettivo quello di fornire la possibilità ai programmi di non considerare la presenza di altri programmi e ha anche come obiettivo quello di semplificare la programmazione dei software.

Quindi l'insieme delle funzioni di gestione del processore, gestione della memoria e gestione delle periferiche di I/O fanno credere ad ogni programma in esecuzione di avere a sua completa disposizione una macchina di von Neumann completa, anche se in realtà le risorse della macchina sono condivise.

Gestione del processore e della memoria centrale

Per la gestione del processore è necessario avere ad esempio delle funzioni per la gestione dei processi: creazione e terminazione dei processi, sospensione e riattivazione dei processi, schedulazione dei processi, sincronizzazione tra processi, gestione di situazioni di stallo (deadlock), comunicazione tra processi.

Un altro insieme di funzioni necessarie sono quelle necessarie per la gestione della memoria centrale: queste funzioni costituiscono il supporto per la multiprogrammazione, devono provvedere alla allocazione e deallocazione della memoria ai processi, devono provvedere al caricamento e allo scaricamento di processi e di loro porzioni in memoria centrale, garantendo la protezione della memoria.

Gestione delle periferiche

Per la gestione delle periferiche si ha bisogno di fornire una omogeneità di interazione: deve consentire la configurazione e inizializzazione, gestione ottimizzata dei dispositivi di ingresso, uscita, memorizzazione di massa e rete informatica, protezione delle periferiche, bufferizzazione e caching.

File System

Il livello di astrazione superiore che rappresenta il File System e che permette l'accesso alle informazioni in modo logico e non fisico presenti nel sistema, deve permettere di creare file come componente elementare che favorisce il contenimento di informazioni e la gestione di questi.

Il file system deve quindi dare all'utente la possibilità di creare e cancellare file e cartelle, leggere e scrivere, copiare, ricercare, proteggere, gestirne l'accounting e permettere il ripristino.

Gestione dell'interfaccia utente

Interfacciarsi con il SO vuol dire fornire quelle funzioni che permettono di dare comandi e ricevere i risultati. Esiste quindi un interprete di comandi per gli utenti con il quale questi interagiscono e un interprete di comandi per i programmi applicativi (che permettono di effettuare le chiamate di sistema).

Devono quindi essere presenti le funzioni: interprete comandi a livello utente e a livello applicativo (chiamate di sistema), librerie di sistema (modo con il quale i comandi possono interagire con il resto del sistema), gestione delle autenticazioni e gestione degli errori e dei malfunzionamenti.

L2 - Architetture dei sistemi operativi

Sistema monolitico

Una primitiva struttura dei sistemi operativi è la cosiddetta 'monolitica'. In questi sistemi sono raccolte tutte le funzioni necessarie alla gestione dell'hardware e dell'interfaccia utente, ma senza una particolare organizzazione se non quella dettata dalle esigenze di programmazione.

Si tratta dunque di un sistema alquanto caotico dove ogni funzione è potenzialmente in grado di richiamare qualunque altra, rendendo così estremamente ardui i compiti di manutenzione e sviluppo del sistema operativo.

Questo genere di architettura è stato usato molto fino agli anni settanta in cui i calcolatori usati come i compiti svolti erano piuttosto elementari. Il sistema è compatto, veloce ed efficiente ed è quindi un approccio adatto a sistemi semplici.

Sistema con struttura gerarchica

Organizzazione delle varie funzioni del sistema operativo in gruppi dipendenti fra di loro. In particolare si sono creati vari livelli funzionali dal più vicino alla struttura hardware al più astratto, più vicino cioè all'interfaccia utente. Si è poi stabilito che ogni livello potesse comunicare esclusivamente con livelli di strato inferiore, anche se non necessariamente con quello immediatamente successivo.

Questa nuova organizzazione ha reso più semplice la manutenzione rispetto ai precedenti sistemi monolitici,ma racchiude in sé ancora dei limiti derivanti in particolare dalla non chiara suddivisione in ruoli delle singole funzioni.

Controllare anche la progettazione risulta un compito difficile, in quanto è necessario stabilire a priori a quale livello deve appartenere la funzione da implementare. Se non viene scelto il livello corretto si rischia di rendere tale funzione inaccessibile ad altre di livello diverso (inferiore) che potrebbero averne bisogno.

Sistema stratificato

Questi introducono una chiara separazione modulare delle funzioni svolte da ciascun componente del sistema operativo. Si parte quindi da un modulo di base per la gestione del microprocessore, su cui poggia il modulo per la gestione della memoria che a sua volta funge da base per il modulo di gestione delle periferiche e così via fino al modulo di interfaccia utente.

In questa maniera le varie componenti sono gerarchicamente suddivise in livelli funzionali, che permettono una più semplice manutenzione del sistema, penalizzando però l'efficienza dell'intero sistema, perdendo tempo nel passaggio e nelle comunicazioni tra un livello e l'altro.

Sistema a microkernel

L'organizzazione interna delle funzioni del sistema operativo è estremamente importante sia per la fase di manutenzione sia per la fase di sviluppo. Si è sentita quindi l'esigenza di approfondire e migliorare ulteriormente tale organizzazione, introducendo una netta separazione tra funzioni atte alla gestione dei meccanismi e quelle atte alla gestione delle politiche.

Per meccanismi si intendono tutte quelle operazioni, fisse, raramente modificabili, di accesso alle risorse, mentre le politiche rappresentano quel genere di definizioni più astratte di diritto, piuttosto che priorità o ordine di uso di una risorsa. Nasce così il concetto di microkernel ovvero a tutti gli effetti un kernel ma scremato di quelle componenti non strettamente necessarie alla gestione dei meccanismi. Tutto il resto, ovvero tutto ciò che è politica, viene organizzato in processi al di sopra del microkernel.

Con questa struttura i programmi utente in esecuzione non comunicano direttamente con i servizi messi a disposizione dal sistema operativo, ma comunicano indirettamente mediante lo scambio di messaggi con il microkernel. Questo è il limite principale del sistema, perché ne riduce le prestazioni. In compenso si ha un ottima modificabilità e la possibilità di estendere il sistema senza modificarne il kernel, inoltre vi è una maggiore affidabilità in quanto la maggior parte dei servizi funziona in modalità utente.

Sistemi a moduli funzionali

Nell'ottica di una sempre più semplice ed efficiente espansione e manutenzione globale del sistema, si è passati all'utilizzo delle moderne tecniche di programmazione ad oggetti. Ciò permette di costruire kernel modulari a cui è possibile aggiungere o togliere dinamicamente delle componenti a seconda delle esigenze. Si torna quindi ad avere un kernel limitato alle funzioni di base intorno al quale vengono agganciati moduli diversi in base alle esigenze del momento.

Questo approccio ha diversi punti in comune con le soluzioni a strati (le interfacce di interazione con i diversi moduli) ed a microkernel (un kernel con funzioni basilari), è però più efficiente in quanto ogni funzione non è vincolata a richiamarne solo di appartenenti a livelli inferiori ed ognuna di esse può comunicare con le altre direttamente, senza scambiare messaggi con il kernel.

Sistema a macchine virtuali

Dai sistemi a strati nasce l'idea di macchine virtuali. Un sistema operativo, attraverso varie tecniche, è in grado di simulare un processore ed una memoria per ogni processo in esecuzione, in modo tale da creare l'illusione di avere una macchina (quindi virtuale) per ogni processo.

Questa tecnica non mette a disposizione nessuna nuova funzionalità, ma fornisce un'interfaccia identica all'hardware sottostante gestita dal kernel di macchina virtuale che è in grado di generare più macchine virtuali su cui installare sistemi operativi diversi. Ottenendo così la convivenza di sistemi operativi eterogenei. Il limite più grande lo si ha nel considerevole calo di prestazioni.

L3 - Generazione e avvio di un sistema operativo

Generare un Sistema Operativo

Generare un SO è una attività complessa; si tratta di individuare le caratteristiche dell'ambiente dove si vogliono fare operare i programmi e gli utenti per quella specifica installazione e quindi capire le caratteristiche degli utenti, dei programmi, i carichi di lavoro che essi generano etc.. per garantire una equa ripartizione dell'uso delle risorse tra i vari utenti e per gestire le risorse in maniera efficiente. E' necessario applicare questi parametri per caricare il codice eseguibile per caricarlo poi in memoria e farlo partire.

Avvio di un Sistema Operativo

I metodi di avviamento (bootstrap) del sistema sono vari e possono essere svolti in uno o più passi e ciò consente di avere diversi gradi di modificabilità ed efficienza del SO. I metodi di avviamento si dividono in: 1 passo, 2 passi, 3 passi.

Bootstrap ad 1 passo

In memoria centrale si considerano due porzioni: RAM (in cui è possibile leggere e scrivere) e ROM (memoria a sola lettura che contiene dei valori permanenti).

Il SO è presente in ROM e questo viene letto immediatamente all'accensione del computer. Essendo il SO salvato nella memoria centrale, il suo caricamento è rapidissimo ma essendo la ROM a sola lettura, il SO non è modificabile. E' utilizzato specialmente nei Sistemi Embedded.

Bootstrap in 2 passi

Riduce la complessità dell'aggiornamento favorendo l'aggiornamento del SO. In questo caso il SO non è tutto nella ROM. Infatti in essa è presente solo un loader del SO. Esso si occuperà nella prima fase di inizializzare l'hardware e reperire su un'altra memoria di massa facilmente modificabile (in una posizione ben determinata) il loader del SO.

Quindi il SO è modificabile mentre il loader non lo è (ROM). La seconda fase consiste nel caricare in memoria centrale il resto del SO dal disco rigido.

Bootstrap in 3 passi

L'avviamento in tre passi vuole superare il limite imposto dal fatto che il SO deve stare in un solo settore specifico del disco (quello memorizzato nel loader).

L'idea è quella di avere in memoria centrale un piccolo loader posto in ROM che provvede a mantenere un riferimento alla porzione del disco rigido che contiene il loader vero e proprio del sistema operativo. Questo verrà posto in RAM dal caricatore di base e sarà questo a sapere dove si trovano le varie porzioni di SO che devono essere caricate in memoria centrale. Il terzo passo consisterà nel caricare il SO in RAM.

In particolare, i passi eseguiti dal bootstrap sono i seguenti:

1. il caricatore elementare viene caricato in memoria ed eseguito. Si trova da qualche parte in ROM.

2. il caricatore elementare carica in memoria il caricatore complesso, non il SO. Questo è il cosiddetto loader di 2° livello. Questo loader può cercare SO ovunque nei dischi, non è limitato al primo settore del primo disco o cose simili
3. infine il caricatore del SO si preoccupa di avviare il SO stesso, ed è possibile passargli parametri per caricare moduli opzionali o simili.

Si usa questo sistema per dare l'opportunità di caricare più sistemi operativi, magari da un menu, o anche quando un SO è abbastanza grande da non starci nel settore iniziale di un disco. Ecco perché serve un caricatore secondario. Il caricatore secondario può anche offrire un'interfaccia all'utente, come LILO o GRUB.

L4 - Interfacce dei sistemi operativi

Interfacce

L'interfaccia utente è costituita dall'interprete dei comandi fornita all'utente (shell) che rimane in attesa di un comando e rimane in una attesa infinita fintanto che l'utente non dichiara di terminare il comando (fetch). Allora l'interprete analizzerà e verificherà la sintassi del comando (decode) e in caso di esito positivo passerà all'esecuzione dello stesso (execute) attivando un processo ausiliario. Al termine dell'esecuzione del processo il controllo ritorna all'interfaccia utente.

L'interazione con l'utente può essere:

- testuale: l'utente deve digitare il comando e i parametri in una shell
- grafica: si basa su icone e menu che permettono di selezionare i comando e i parametri, minimizzando la possibilità di effettuare errori di sintassi/errori di digitazione.

Interfacce programmative

L'interfaccia programmativa permette ai programmi di effettuare delle chiamate di sistema (system call) il cui obiettivo è quello di proteggere il sistema operativo, garantire l'integrità dei dati, garantire l'esecuzione completa delle procedure di sistema operativo e garantire l'esecuzione dei controlli nelle procedure di sistema.

Le risorse vanno gestite bene, e a questo scopo è importante cercare di proibire ad un programmatore di bypassare il SO per accedere direttamente all'hardware.

I vantaggi di chiamare il SO invece di accedere direttamente all'hardware sono molteplici:

- il programmatore non deve occuparsi dei dettagli di 1000 tipi di hardware diverso;
- non si devono reimplementare ogni volta le stesse funzioni;
- se le funzioni sono complesse, si può sbagliare a scriverle, se non si è esperti in quel ramo.

Le chiamate di sistema non sono normali chiamate di procedura, anche se concettualmente lo sono. Ogni volta che ne viene fatta una, si controllano ben bene i permessi di chi le sta chiamando.