Architettura dei Sistemi Operativi

Struttura dei Sistemi Operativi

Funzioni di un Sistema Operativo

Obiettivi di un Sistema Operativo

I principali obiettivi di SO sono:

- **Astrazione**: innalzare il livello di astrazione dei componenti del sistema di elaborazione semplificandone ed ottimizzandone le attivita';
- **Virtualizzazione**: creare un'immagine del sistema di elaborazione dedicata a ciascun programma in esecuzione (indipendenza tra i programmi in esecuzione).

Organizzazione logica di un Sistema Operativo

Tramite l'astrazione e la virtualizzazione, ogni programma P che viene eseguito dal calcolatore *crede* di avere a disposizione la CPU, la memoria e le periferiche I/O tutte per se. Questo e' ottenuto da delle componenti SW all'interno del SO:

- **Gestione del processore**;
- **Gestione memoria centrale**;
- **Gestione I/O**.

L'astrazione ottenuta richiede pero' che ciascun programma *conosca* la struttura fisica del sistema e dove sono poste (fisicamente) le varie informazioni necessarie per poter essere eseguito. Il SO, dunque, fornisce ai programmi una visione ancora piu' astratta dell'HW in cui le informazioni hanno un riferimento logico, questo avviene tramite la **Gestione del FileSystem**.

L'**Interfaccia untenti** e' un insieme di funzioni che connette gli utenti fisici e gli applicativi al sistema stesso.

Funzioni di un Sistema Operativo

Ogni SO mette a disposizione funzioni per realizzare la gestione dei singoli componenti della macchina di von Neumann e per gestire il livello logico di astrazione per il reperimento delle informazioni costituito dal FileSystem e l'interfaccia utente.

Per la gestione del processore e' necessario avere delle funzioni che gestiscano i programmi in esecuzione (**processi** - sequenze di attivita' svolte dal sistema):

- Creazione e terminazione;
- Sospensione e riattivazione;
- Schedulazione;
- Sincronizzazione tra processi;
- Gestione di situazioni di stallo (**deadlock**);
- Comunicazione tra processi.

Per la gestione della memoria centrale servono delle funzioni che individuino la porzione di memoria contenente le informazioni necessarie al processo per operare:

- Supporto per la multiprogrammazione;
- Allocazione e deallocazione della memoria per i vari processi;
- Caricamento e scaricamento di processi e di loro porzioni in memoria centrale;
- Protezione della memoria centrale.

Per la gestione delle periferiche abbiamo bisogno di funzioni che forniscano un'omogeneita' di interazione:

- Configurazione ed inizializzazione;
- Interfaccia generale ed omogenea;
- Gestione ottimizzata dei dispositivi di I/O, memoria di massa e rete informatica;
- Protezione delle periferiche;
- Meccanismi di buffering e di caching.

Il livello di astrazione ulteriore fornito dal Filesystem, che consente l'accesso alle informazioni in modo logico e non piu' fisico, deve permettere:

- Individuazione dell'albero dei file e delle directory;
- Creazione e cancellazione dei file;
- Lettura e scrittura;
- Copiatura;
- Ricerca;
- Protezione e sicurezza;
- Accounting;
- Salvataggio e ripristino.

L'interfaccia utente e' quello strato in cui gli utenti fisici e i programmi possono dare comandi al calcolatore e riceverne i risultati:

- Interprete dei comandi a livello utente;
- Interprete dei comandi a livello programmi (chiamate a sistema);
- Libreria di sistema;
- Autenticazione degli utenti e dei processi per garantire l'accesso alle risorse;
- Gestione degli errori e malfunzionamenti.

Generazione e avvio di un Sistema Operativo

Generare il sistema operativo significa trovare la configurazione del sistema e andarla ad applicare, mentre avviare il sistema operativo significa caricarlo in memoria centrale all'accensione del sistema e fare in modo che prenda il controllo dell'archiettura di elaborazione.

Generazione del Sistema Operativo

Si tratta di un'attivita' molto complessa che consiste nel:

- **Identificare le caratteristiche** dell'ambiente in cui vogliamo far operare i programmi e gli utenti per quella specifica

installazione, dunque capire per gli utenti e per i programmi:
    - Quali sono le caratteristiche;
    - Il carico di lavoro che generano;
    - Le richieste di uso delle risorse che essi generano.
- **Definire i parametri** ottimali del sistema operativo per gestire le risorse in maniera efficiente per garantire un'equa ripartizione dell'uso delle risorse tra le varie tipologie di utenti;
- **Applicazione dei parametri** per la generazione del nuovo codice eseguibile del sistema operativo;
- Aggiornare il sistema operativo nel sistema di elaborazione con la nuova configurazione applicata.

Identificazione della caratteristiche

E' un'attivita' che richiede:

- Un'accurata analisi delle applicazioni che si vogliono utilizzare;
- Delle caratteristiche di carico di lavoro sul processore;
- Dell'uso delle risorse sul sistema;
- Analizzare l'ambiente (composto da applicativi e utenti) per comprendere le loro abitudini e dunque i rispettivi carichi di lavoro;
- Valutare le caratteristiche del carico di lavoro consiste nel mettere insieme tutte le informazioni raccolte sulla *modalita' d'uso* e la *modalita' di funzionamento* degli applicativi per creare un **modello** che descriva il carico di lavoro e richieste per le singole risorse fisiche o informative del sistema;
- La raccolta delle informazioni, che puo' essere effettuata in modo manuale o automatico (monitorando il comportamento degli utenti in un ambiente simulato, oppure ponendosi nell'ambiente reale e osservando come gli utenti lavorano);
- Infine, la valutazione delle caratteristiche dello scenario operativo puo' essere fatto in base all'esperienza o in base statistica.

Definizione dei parametri

Per definire i parametri e' necessario:

- Analizzare il modello costruito dall'analisi del carico di lavoro;
- Valutare quali sono i migliori parametri che portano il sistema operativo a comportarsi il meglio possibile rispetto alle richieste degli utenti e delle applicazioni che essi lanciano. Puo' essere fatto:
    - In maniera manuale in base all'esperienza o in base alla statistica;
    - In maniera automatica in base a casi ben predefiniti o in base a delle regole preimpostate.

Applicazione dei parametri

Una volta definiti i parametri di configurazione del sistema operativo:

- Vanno applicati modificando i file di configurazione del sistema operativo;
- Si eseguono le procedure che generano del codice eseguibile per i moduli che sono stati modificati in fase di configurazione;
- Si genera il codice complessivo per tutti il sistema operativo e di tutti i programmi di sistema realtivi.

Aggiornamento del sistema

Consiste nel:

- Memorizzare la nuova versione del sistema operativo e dei programmi di sistema;
- Caricamento del nuovo sistema operativo in memoria centrale.

Avviamento del sistema operativo

E' un'operazione che puo' essere eseguita all'accensione del sistema in vari modi a seconda del supporto HW che viene fornito. Il metodo di avviamento (boostrap) puo' essere effettuato:

- In singolo passo;
- In due passi;
- In tre passi.

Questo consente di ottenere diversi gradi di modificabilita' del sistema operativo e conseguentemente diversi gradi di efficienza nell'accesso alle funzioni di sistema in fasi di esecuzione o avviamento.

Avviamento in singolo passo

In memoria centrale consideriamo due porzioni:

- Una parte e' realizzata con una RAM;
- Un'altra e' realizzata con una ROM che contiene i valori memorizzati anche dopo lo spegnimento, duqnue ideale per contenere il codice del sistema operativo.

Il *boostrap primario* consiste nel caricamento rapido del sistema operativo e delle funzioni di sistema, in quanto tutto e' gia' in memoria centrale. Lo svantaggio e' che il sistema operativo non e' modificabile a meno che non venga sostituito il dispositivo ROM.

Avviamento in due passi

Un'altra tecnica riduce la complessita' dell'aggiornamento permettendo una modificabilita' piu' semplice del sistema operativo.

L'idea consiste nel non mettere in memoria centrale tutto il sistema operativo in modo fisso, ma di avere nella ROM solo il *loader* del sistema operativo nella fase di *bootstrap primario*.

Successivamente, nella fase di *boostrap secondario*, il *loader* reperira' su un'altra memoria di massa (in una posizione predeterminata) il sistema operativo vero e proprio. Infine quando il sistema operativo sara' caricato in RAM, il *loader* cedera' ad esso il controllo del calcolatore.

Avviamento in tre passi

L'avviamento in ulteriori passi successivi si pone come obiettivo di superare il limite della reperibilita' del sistema operativo, il quale deve essere in un solo settore specifico nella memoria di massa esterna.

In ROM si pone un *base loader* che provvede a mantenere il riferimento ad una porzione predeterminata del disco esterno in cui si trova il *loader* vero e proprio.

Una volta caricato il *loader* in RAM questo si occupera' di reperire le varie porzioni del sistema operativo sparse nel disco esterno.

Il terzo passo consistera' nel caricare in RAM il sistema operativo vero e proprio.

Un caricamento puo' essere reso particolarmente complesso andando a caricare porzioni/moduli del sistema operativo solo quando richiesto per eseguire le specifiche funzioni richieste dagli ambienti applicativo.

Avviamento in n passi

Avviare il sistema operativo con ulteriori passi fornisce un alto livello di modificabilita' del sistema operativo ma comporta inevitabilmente un caricamento piu' lento e complesso. L'accessibilita' alle funzioni e' dipendente dal modulo caricato in RAM.

Interfacce dei sistemi operativi

Interfaccia verso l'utente

E' costituita dall'interprete dei comandi, la **shell**, che:

- Rimane inizialmente in attesa che l'utente digiti il comando desiderato (**fetch**) finche' l'utente non dichiara il completamento dell'immissione del comando;
- Provvede ad analizzare e verificare la correttezza del comando (**decode**);
- Esegue (**execute**) il comando attivando un processo ausiliario (che esegue effettivamente il comando). Al termine del processo ausiliario, il controllo ritorna all'interfaccia utente.

L'interazione con l'utente puo' avvenire fondamentalmente in due modalita':

- **Text Interface**: inserimento del comando tramite tastiera e pressione del Carriage Return;
- **Graphical User Interface**: inserimento del comando semplificato tramite icone e menu', limitando la digitazione e riducendo di conseguenza gli errori di digitazione.

Interfaccia verso i programmi

E' un'interfaccia che permette ai programmi di richiedere le funzioni fornite dal sistema operativo, chiamate **system call** (esistono anche le *supervisor call* e le *monitor call*).

L'obiettivo di questa interfaccia e della chiamata di sistema e' quella di proteggere il sistema operativo, garantendo:

- L'integrita' dei dati;
- La completa esecuzione delle procedure di sistema operativo;
- Che gli eventuali controlli previsti dalle procedure vengano effettivamente eseguiti.

Quindi la chiamata di sistema e' molto simile ad una chiamata di procedura, ma effettivamente non lo e'.

System Call

Una chiamata di sistema effettuata da un programma viene eseguita in molteplici passi:

- Il programma inserisce nello stack i parametri e le opzioni della chiamata che vuole effettuare (a seconda della libreria utilizzata l'ordine delle parole sullo stack puo' variare);
- Il programma richiede l'esecuzione della chiamata di sistema;
- La chiamata di sistema inserisce il numero identificativo della chiamata in un registro del processore, dopodiche' esegue un'istruzione **TRAP** per passare dalla *modalita' utente* alla *modalita' kernel*;
- La chiamata di sistema inizia la sua esecuzione ad un indirizzo fisso all'interno del kernel;
- Il codice del kernel comincia ad esaminare il numero di chiamata di sistema, quindi la smista al corretto gestore della

chiamata di sistema (attraverso una tabella di puntatori a gestori delle chiamate di sistema, indicizzata dal numero della chiamata);

- Viene eseguito il gestore della chiamata di sistema e il risultato ottenuto viene utilizzato come risposta all'interruzione TRAP lanciata precedentemente;
- Viene restituito il controllo alla *modalita' utente* all'istruzione successiva alla TRAP e si ritorna al programma utente in modo usuale;
- Per completare il lavoro, il programma utente deve liberare lo stack.