Con il termine Sistema operativo si fa riferimento all'insieme dei moduli software di un sistema di elaborazione dati dedicati alla sua gestione.

Compito fondamentale di un S.O. è infatti la gestione dell'esecuzione dei lavori (Job) sottoposti dagli utenti (software applicativo) utilizzando "al meglio" tutte le risorse del sistema (processi, memoria, unità di I/O, files, etc.) necessarie all'avanzamento dei lavori stessi.

In tale contesto il Sistema Operativo sviluppa le funzioni di interfaccia tra:

- Il software applicativo e l'hardware
- L'operatore ed il sistema di elaborazione.

I compiti del S.O. si possono riassumere in:

- accettare le richieste dell'utente;
- eseguire i processi
    - gestendo le interazioni fra processi
    - assegnando loro il processore
    - assegnando le altre risorse di cui hanno bisogno
- permettere lo sviluppo di nuovi programmi

delle richieste dell'utente si occupa la shell, l'interfaccia con l'utente.

Per lo sviluppo di nuovi programmi il S.O. mette a disposizione vari programmi di utilità come editor, compilatori, linker ecc.

Il nucleo si occupa della gestione del processore e delle interruzioni. Le funzioni di gestione delle risorse sono svolte da processi separati. I gestori delle risorse generalmente presenti in un S.O. sono:

- il gestore della memoria
- il gestore delle periferiche
- il gestore delle informazioni (file system).

Ogni volta che si accende il computer, il nucleo del S.O. viene caricato, mediante il bootstrap nella memoria RAM.

Appartengono al nucleo del S.O.

- I moduli per la gestione della CPU
- I moduli per la gestione delle interruzioni

Le funzioni del nucleo sono:

- creare e cancellare processi;
- mantenere aggiornato lo stato dei processi e della CPU
- allocare la CPU in base ad una specifica politica di gestione
- rilasciare la CPU
- gestire la sincronizzazione tra i vari processi

Un processo è un'attività elaborativa che è sviluppata sequenzialmente in un sistema in concorrenza o in parallelo con altre attività sotto il controllo di un gestore (S.O).

In altri termini costituisce un processo un qualsiasi segmento di programma in corso di esecuzione a cui il S.O. possa attribuire le risorse del sistema e che possa essere eseguito in concorrenza o in parallelo con altri segmenti dello stesso o di altri programmi.

Poiché un processo è una sequenza di istruzioni in corso di esecuzione, al concetto di processo è associato il concetto di stato.

Con riferimento alla risorsa processore si definiscono stati di avanzamento di un processo:

RUN (corrente): il processo possiede la risorsa processore che esegue le sue istruzioni

WAIT (attesa): il processo non possiede la risorsa processore e non la richiede essendo in attesa di un evento.

READY (pronto): il processo non possiede il processore ma è pronto per usufruirne.

Affinché il S.O. possa gestire un processo è necessario che ad esso sia dedicata parte della struttura dati del S.O. in cui siano memorizzate tutte le informazioni necessarie alla sua gestione. Ogni volta che un processo lascia la CPU, viene salvato in memoria una descrizione del processo (descrittore) per poter riprendere successivamente l'esecuzione dal punto in cui era stata interrotta. In particolare viene salvato il contenuto dei registri che viene ripristinato quando il processo riprende la CPU (TASK SWITCH).

La descrizione del processo è detta PCB (Process Control Block); un PCB contiene:

- PID: un numero progressivo identificatore del processo;
- lo stato in cui si trova: run, ready, wait;
- il contenuto dei registri;
- l'indirizzo in memoria centrale di dove si trova il processo;
- le periferiche usate;
- i file aperti.

Ogni processo ha un proprio PCB e nessun processo ha accesso al PCB degli altri.

Affinché un Job di utente possa essere eseguito è necessario da esso nasca almeno un processo a cui il S.O. possa attribuire risorse. Questi processi devono poi essere eliminati quando le loro istruzioni sono state eseguite e in ogni caso quando il Job termina.

Ne deriva che un processo è:

-Creato

-Distrutto

Quando viene eseguita l'ultima istruzione, il processo passa nello stato di fine, il suo PCB viene eliminato e il S.O. dichiara libera la memoria occupata.

I processi in esecuzione contemporaneamente possono appartenere a programmi utente o al sistema operativo. Il codice del S.O. viene eseguito in modalità privilegiata ed ha accesso a tutta la memoria e

all'hardware di sistema. Le applicazioni vengono eseguite in modalità non privilegiata (modalità utente)ed hanno un accesso limitato alla memoria e all'hardware, attraverso richieste di servizio al S.O. Le funzioni di gestione delle risorse sono svolte da **processi server (o gestori delle risorse**). Il processo server riceve le richieste di assegnazione o rilascio di una risorsa da parte di un processo client ed esegue per loro conto le **funzioni di gestione**. Per richiedere o rilasciare risorse, i processi client comunicano con il server tramite messaggi.

Schedulazione del processore.

Quando il processore viene rilasciato da un processo deve essere assegnato ad un altro processo in stato di pronto. La scelta del processo da mandare in esecuzione viene determinata da un algoritmo di schedulazione del processore.

Il S.O. gestisce il rilascio del processore in maniera **preempitive,** ossia il S.O. stabilisce un limite di tempo di utilizzazione del processore per ogni processo. Scaduto il tempo, il S.O. forza il processo a rilasciare il processore. In genere la schedulazione avviene assegnando una priorità ai processi che ne determina l'ordine di esecuzione. Ad ogni processo viene assegnato un numero che rappresenta il suo livello di priorità. I processi pronti (in stato di ready) vengono inseriti in diverse code corrispondenti ai vari livelli di priorità.

Un thread è una parte di processo che viene creata automaticamente quando si crea un processo (thread principale). Ogni thread può creare a sua volta altri thread. Un thread condivide tutto il codice del processo ma ha una sua propria area dati. Lo switch tra i thread è quindi molto veloce.

Gestione delle interruzioni.

L'interrupt è un meccanismo mediante il quale vengono comunicati alla CPU alcuni eventi ben precisi (scadenza del time slice, errore nel programma…). Un interrupt è un segnale hardware che viene inviato dai vari dispositivi alla CPU. Quando invece un processo richiede al S.O. un'operazione di I/O genera un'interruzione di tipo software.

Quando si verifica un interrupt, il S.O. sospende l'esecuzione del processo e attiva una procedura specifica per l'interruzione richiesta. La routine di gestione dell'interrupt può essere attivata solo quando la CPU ha completato il ciclo di esecuzione di un'istruzione. La parte di S.O. che si occupa della gestione delle interruzioni si chiama interrupt handler. Può succedere che, mentre è in esecuzione la routine di gestione dell'interrupt arrivi alla CPU un altro segnale di interruzione. In questo caso si usa il metodo delle interruzioni vettorizzate: ad ogni tipo di interrupt è assegnato un livello di priorità ed è possibile interrompere una routine di interrupt solo se il nuovo interrupt ha una priorità maggiore.

Per gestire un interrupt il S.O. deve:

1. salvare il PCB del processo in esecuzione
2. eseguire la routine di gestione dell'interruzione
3. ritornare all'esecuzione del processo sospeso riattivando il suo PCB

WINDOWS è un sistema operativo multithreading di tipo preempitive. L'unità di schedulazione è il thread, la durata del time-slice è variabile e si può cambiare dal pannello di controllo. Ogni thread ha una priorità

identificata con un numero che va da 1 a 31. I thread delle applicazioni utenti usano le priorità da 1 a 15. Si possono avere informazioni sui processi in esecuzione attraverso il Task Manager.

Windows è realizzato secondo il modello client/server. Le applicazioni eseguite dall'utente sono i client che richiedono i servizi dei moduli del S.O. che sono i server. L'approccio client/server si risolve in un sistema operativo modulare; i server sono piccoli e indipendenti. L'ambiente grafico è strettamente legato nel sistema.

Definizione di risorsa

E' una risorsa un qualsiasi componente di un sistema di elaborazione dati che può essere usato da un processo per il suo avanzamento e la cui disponibilità può quindi condizionarne l'evoluzione.

Esempi di risorse sono:

- I processori (CPU, canali)
- Le memorie (Centrali e di massa) intese come insiemi di elementi indirizzabili atti alla memorizzazione delle informazioni (locazioni, settori di disco, etc.)
- I meccanismi di indirizzamento delle memorie la cui disponibilità consenta l'accesso agli elementi indirizzabili:
- I file
- I messaggi

Classificazione delle risorse

Esistono diverse classificazioni possibili:

A) Hardware
   Software

B) Seriale
   Non seriale (shared)

C) Permanente
   Consumabile

Esempi: Una CPU è una risorsa hardware-seriale-permanente.

Una biblioteca di programmi è una risorsa software-non seriale-permanente.

Un archivio dati soggetto ad aggiornamento è una risorsa software-seriale-permanente.

Un messaggio è una risorsa software-seriale-consumabile.

Per le risorse seriali si definisce il:

periodo seriale: intervallo di tempo minimo in cui la risorsa è assegnata ad un processo.

Per individuare il periodo seriale di una risorsa bisogna tener presente i tipi possibili di assegnazione della risorsa ai processi:

A) Statica: la risorsa è assegnata al processo nel momento in cui è creato e il processo la rilascia quando viene distrutto (eliminato).
Dinamica: la risorsa è assegnata al processo quando è richiesta; il processo la rilascia quando ha terminato di utilizzarla. Una risorsa assegnata dinamicamente deve essere assegnata e rilasciata più volte durante l'avanzamento di un processo.

B) Con prerilascio
Senza prerilascio

Per prerilascio si intende la sottrazione di una risorsa ad un processo prima che sia terminato (se assegnata staticamente) o prima che abbia terminato di utilizzarla (se assegnata di dinamicamente)

Sono pertanto possibili assegnazioni:

- Statica senza prerilascio
- Statica con prerilascio
- Dinamica senza prerilascio
- Dinamica con prerilascio

Per tutte le risorse assegnate staticamente senza prerilascio il periodo seriale è la durata del processo ed è quindi indipendente dalla risorsa.

Per le risorse assegnate con altre modalità di assegnazione il periodo seriale dipende dalla risorsa.

Esempi:

Per i processori (CPU) la cui assegnazione è sempre dinamica, il periodo seriale (teorico) è la durate dell'esecuzione di una istruzione.

Per i file soggetti ad aggiornamenti (assegnati dinamicamente senza prerilascio) il periodo seriale è la durata dell'aggiornamento da parte di un processo.

Il periodo seriale del circuito di indirizzamento di un disco (assegnato dinamicamente senza prerilascio) è pari alla durata dell'accesso ad un settore.

Anche alle risorse il S.O. dedica parte della propria struttura dati per memorizzare le informazioni necessarie alla loro gestione.

A ciascuna risorsa è infatti associato un descrittore di risorsa.

La struttura del descrittore varia da risorsa a risorsa.

Esempi:

Il descrittore di un'area di memoria di un disco gestito a partizioni fisse contiene:

- Indirizzo base della partizione, lunghezza, indirizzo del descrittore del processo che impegna la partizione.

Il descrittore di un canale di input/output (I/O) contiene tra l'altro:

- Lo stato del canale.
- L'indirizzo del descrittore del processo che impegna il canale.
- Gli indirizzi dei descrittori dei circuiti di indirizzamento della memoria di massa collegata al canale.
- Gli indirizzi dei descrittori delle aree di memoria di massa accessibili dal canale.

I concetti di processo e di risorsa sono di fondamentale importanza per lo studio dei S.O. e possono subito essere impiegati per una sua descrizione.

In realtà un S.O. può essere visto secondo diverse ottiche tra di loro complementari. In particolare lo si può descrivere:

-**Da un punto di vista statico- funzionale**: il S.O. è un insieme di programmi dedicati alle risorse di un sistema che sono accessibili da una moltitudine di utilizzatori; "**S.O. visto come gestore di risorse"**

-**Da un punto di vista dinamico** secondo un modello che evidenzia come un job di utente attraversa un sistema utilizzando le sue risorse e attivando via via i segmenti del S.O. dedicati alla loro gestione; **"S.O. visto come gestore di processi**|.

Il S.O. come gestore di risorse

Il S.O. è visto come un insieme di programmi gestori di risorse.

Ciascun gestore sviluppa le funzioni di:

- Memorizzare lo stato delle risorse gestite;
- Applicare una politica di assegnazione della risorsa ai processi (a chi assegnarla, quando assegnarla, per quanto tempo);
- Allocare la risorsa ai processi secondo la politica adatta;
- Recuperare la risorsa.

Con riferimento alla risorsa gestita si parla di:

- funzioni di gestione della memoria;
- funzioni di gestione del processore; i programmi relativi sono detti traffic controller, processo scheduler, dispatcher;
- funzioni di gestione dei device di I/O; i programmi relativi sono detti I/O traffic controller, I/O scheduler;
- funzioni di gestione delle informazioni (file) i programmi relativi sono detti file system create/delete, open/close.

Il S.O. visto come gestore di processi

Il S.O. è visto come insieme di programmi che, attivati via via a partire dalla sottomissione di un job:

- lo memorizzano sul disco (spoolin);
- ne schedulano l'esecuzione;

- cercano i processi associati;
- ne gestiscono l'avanzamento e le interazioni;
- li distruggono alla terminazione;
- provvedono alla presentazione dei tabulati di uscita (spool-out);

I processi sono in competizione per l'uso delle risorse e quindi interferiscono tra di loro (più processi interferiscono tra di loro se dipendono l'uno dall'altro per l'uso di una risorsa, ma sarebbero altrimenti indipendenti).

Il sistema operativo deve assegnare le risorse ai processi in modo che vengano usate in mutua esclusione;

Il gestore della memoria

Il memory manager gestisce assegnazioni e rilasci della memoria ai processi. Per il caricamento statico assegnazione e rilascio avvengono solo alla generazione e terminazione dei processi. Se si vogliono avere in memoria centrale più processi in contemporanea, allora la memoria deve essere una risorsa condivisa. La dimensione della memoria limita sia il numero di processi che possono coesistere sia la velocità di avanzamento.

Il gestore della memoria deve gestire la memoria in modo da:

- non limitare la possibilità di creare altri processi;
- consentire l'avanzamento di tutti i processi;
- ridurre il numero e la durata delle sospensioni per aspettare il caricamento di codice e dati per poter avanzare.

La coesistenza di aree di memoria assegnate a diversi processi impone meccanismi **di protezione**, cioè bisogna controllare che ogni processo acceda soltanto alla sua area di memoria assegnata, senza invadere aree assegnate ad altri processi o al sistema operativo. Deve comunque essere permesso di condividere aree di codice e dati.

La condivisione dei dati è fondamentale per la programmazione concorrente e causa interazioni tra i processi. La condivisione di codice non causa interazioni tra processi e permette di ottimizzare l'uso della memoria. La condivisione è facilitata nel modello di caricamento a segmentazione, che permette la condivisione di codice e dati.

Ogni programma per essere eseguito deve essere caricato nella memoria RAM, occupando uno **spazio fisico** cioè ad ogni indirizzo logico delle sue istruzioni e dei suoi dati deve essere assegnato un indirizzo fisico nella RAM. Tale operazione è detta **rilocazione** che può essere statica se effettuata al momento del caricamento o dinamica se effettuata durante l'esecuzione usando meccanismi del processore.

L'allocazione della memoria è la tecnica con cui il S.O. concede e assegna memoria.

Nella allocazione **statica** il programma è caricato tutto in memoria all'inizio e risiede in memoria per tutto il tempo dell'esecuzione.

Nella **allocazione dinamica** lo spazio fisico può essere ridefinito più volte e il programma viene caricato e scaricato più volte.

Vediamo quali sono i modi di gestire la memoria con l'allocazione statica ma rilocazione dinamica:

partizione fissa: lo spazio di memoria viene suddiviso in parti di dimensione diverse dette partizioni. La suddivisione in partizioni viene effettuata dall'amministrazione del sistema al momento dell'avvio del S.O. Ogni programma viene caricato in una partizione dove rimane per tutta l'esecuzione; se il programma è piccolo rispetto alla partizione viene sprecato spazio (frammentazione interna). Per la protezione basta controllare che ogni accesso alla memoria sia interno alla partizione del processo stesso.

Partizioni variabili: ogni programma viene caricato in una partizione che occupa fino alla fine dell'esecuzione. La differenza è che le partizioni vengono create dal S.O. in base ai programmi che devono essere eseguiti. All'inizio tutta la memoria disponibile è vista come un'unica partizione libera. Alla richiesta di esecuzione di un processo il S.O. assegna al processo una partizione di dimensione esatta. Quando le partizioni vengono rilasciate dai processi si creano in memoria delle aree libere frammiste ad aree occupate, si ha quindi una frammentazione esterna cioè la creazione un po' alla volta di partizioni piccole e inutilizzabili. Questo fenomeno però è reversibile e può essere risolto mediante un compattamento della memoria. Con questa operazione tutti i programmi presenti in memoria devono essere spostati per poter formare un'unica area libera. Per poter effettuare la compattazione è necessaria la rilocazione dinamica degli indirizzi fisici.

Il gestore della memoria sceglie la partizione da assegnare tra le partizioni libere. La scelta delle partizioni libere può essere fatta con la strategia best-fit o first-fit.

La strategia first fit assegna al processo la prima partizione libera abbastanza grande per contenerlo. La strategia best-fit assegna al processo la partizione più piccola possibile che possa contenerlo, in modo che le partizioni più grandi non vengano suddivise inutilmente.

La rilocazione è dinamica e usa un registro base in cui, al momento dell'esecuzione, viene caricato l'indirizzo iniziale della partizione; all'indirizzo della partizione viene sommato l'indirizzo logico (offset).

Viene usata la stessa tecnica di protezione della partizione fissa.

Segmentazione è adatta per il caricamento di programmi con una struttura logica a segmenti (un segmento può essere una libreria usata dal programma, l'area dati o l'area istruzioni). La segmentazione è un'allocazione statica non contigua cioè un programma può essere allocato in aree separate e non consecutive della memoria fisica

Ogni segmento viene caricato in una partizione col metodo delle partizioni variabili, si ha una riduzione della frammentazione. la segmentazione si presta in modo naturale alla condivisione di codice e dati. In genere i processi condividono segmenti di codice, in modo da avere una sola copia in memoria, e usano segmenti privati per i dati. La rilocazione è sempre dinamica. Si usano dei registri base con l'indirizzo iniziale dei segmenti; in genere si hanno registri base per un segmento di codice e alcuni segmenti dati

Paginazione usa il blocco come unità di assegnazione; il programma viene diviso in pagine della stessa dimensione dei blocchi, ogni pagina può essere allocata anche in un blocco non contiguo. La rilocazione è sempre dinamica.

Il caricamento dinamico ridefinisce lo spazio fisico di un processo più volte durante la vita del processo. I **processi permanenti del S.O. non vengono mai caricati in modo dinamico**. Caricamenti e scaricamenti

(swapping) sono decisi dal S.O. e sono invisibili ai processi; possono riguardare l'intero programma o porzioni di programma (segmenti o pagine).

Quando il caricamento è dinamico oltre allo spazio fisico deve essere assegnato al processo anche uno spazio su disco in un'apposita area (area di swap).

Il meccanismo di swapping rende possibile generare nuovi processi anche quando non c'è memoria sufficiente; basta che sia disponibile l'area di swap.

I processi in esecuzione usano oltre alla memoria centrale anche l'area di swap su disco dando l'impressione di avere una memoria molto più grande di quanto sia in realtà (memoria virtuale).

La memoria virtuale è costituita dalla memoria reale e dall'area di swap su disco.

Per la gestione dell'area di swap su disco, Windows usa dei file chiamati **file di paginazione**. Può essere creato un file di paginazione su ogni disco. La dimensione del file di paginazione viene determinata automaticamente in base alla dimensione della RAM e allo spazio su disco, ma può essere modificata; nella sezione Memoria virtuale si possono impostare le dimensioni minima e massima del file.