# CAPITOLO 3 : I PROCESSI

## 1.CONCETTO DI PROCESSO

Una questione importante che sorge nell’analisi dei S.O. è la definizione delle attività della CPU. Un sistema a lotti esegue job mentre un sistema a partizione del tempo esegue task.

Tali attività sono molto simili in molti aspetti e pertanto vengono definite processi.

Un processo informalmente è un programma in esecuzione.

Un **processo** è un’entità attiva. Ossia è un programma in esecuzione.

Un programma è un’entità passiva, è costituito da un file binario residente nel disco che contiene il codice. Un programma diventa un processo quando viene caricato in memoria ed attende di essere eseguito dalla CPU. Un processo si può trovare in vari stati:

* Nuovo: il processo viene creato
* Attesa:il processo attende che si verifichi qualche evento
* Esecuzione: il processo si trova nella CPU ed è eseguito
* Pronto: il processo è pronto per essere assegnato alla CPU.
* Terminato: il processo ha terminato la sua esecuzione.

Per tener traccia delle informazioni inerenti un processo, ogni processo viene rappresentato dal PCB. È un blocco che contiene molte informazioni sul processo quali il suo **stato**,**informazioni sullo scheduling della CPU**, i **registri della cpu**, il **program counter, informazioni sulla gestione della memoria**, **informazioni sulle risorse utilizzate dal processo** **e informazioni sullo stato di I/O**.

## 2.SCHEDULING DEI PROCESSI

L’obiettivo della multiprogrammazione consiste nel disporre dell'esecuzione contemporanea di alcuni processi in modo da massimizzare l’utilizzo della cpu . L’obiettivo della partizione del tempo è di commutare l'utilizzo della CPU tra vari processi così frequentemente che gli utenti possano interagire con ciascun programma mentre è in esecuzione.

Per raggiungere questi obiettivi , lo scheduler dei processi seleziona dalla coda dei processi un processo il quale viene posto in esecuzione nella cpu. In un sistema mono-processore non vi sarà mai più di un processo in esecuzione ; gli altri processi dovranno attendere finchè la CPU sia nuovamente disponibile.

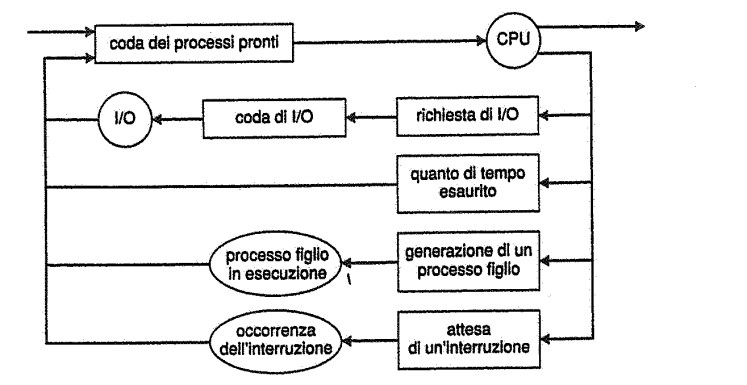
Ogni processo è inserito nella coda dei processi, composta da tutti i processi del sistema. I processi presenti in memoria centrale che sono pronti e nell’attesa di essere eseguiti, vengono collocati nella coda dei processi pronti. Quando viene assegnato ad un processo la cpu, questo viene tolto dalla coda dei processi pronti ed eseguito finche non si verifichi o un’interruzione, o una richiesta di I/O oppure esso termini la sua esecuzione. Poichè nel sistema esistono molti processi, il disco può essere occupato con una richiesta di I/O di un altro processo. L'elenco dei processi che attendono la disponibilità di un dispositivo di I/O si chiama coda del dispositivo.

Un nuovo processo inizialmente si colloca nella coda dei processi pronti, dove attende finchè non è selezionato per essere eseguito (Dispatched). Una volta che gli viene assegnata la cpu ed è nella fase di esecuzione , si può verificare uno dei seguenti eventi:

-il processo richiede una operazione di IO e quindi viene inserito nella coda dei processi del dispositivo;

-il processo può creare un nuovo processo ed attenderne la terminazione;

-il processo può essere rimosso forzatamente dalla cpu a causa di un’interruzione ed essere inserito nella coda dei processi pronti.



## Scheduler

Nel corso della sua esistenza, un processo si trova in varie code di scheduling. Il SO attraverso lo scheduler seleziona i processi dalle suddette code. Spesso , nei sistemi a lotti , accade che più processi vengono sottoposti in esecuzione di quanti se ne possono eseguire. Quindi questi processi vengono trasferiti in memoria secondaria, dove si tengono fino al momento della loro esecuzione(spooling).

Lo scheduler a lungo termine provvede a scegliere questi lavori e a caricarli in memoria centrale dove attendono affinchè siano eseguiti.

Lo scheduler a breve termine (o Scheduler della CPU) invece fa la selezione tra i processi pronti per l'esecuzione e assegna la CPU a uno di loro.

Questi due scheduler si differenziano maggiormente per la frequenza in cui queste operazioni vengono effettuate, lo scheduler a breve temine seleziona processi molto velocemente, nell’ordine dei millisecondi mentre per lo scheduler a lungo termine possono passare anche diversi minuti prima che selezioni un nuovo processo. È importante che lo scheduler a lungo termine faccia un’accurata selezione dei processi che possono essere processo con prevalenza di IO (IO bounds) e processo con prevalenza di esecuzione (CPU bounds). Inoltre lo scheduler a lungo termine controlla il grado di multi-programmazione del sistema.

Il altri sistemi operativi, come quelli a partizione del tempo, è stato introdotto un livello di scheduling intermedio. L’idea alla base di un tale scheduler è che a volte può essere vantaggioso

eliminare processi dalla memoria (e dalla contesa attiva per la CPU), riducendo il grado di multiprogrammazione del sistema. In seguito, il processo può essere reintrodotto in memoria, in modo che la sua esecuzione riprenda da dove era stata interrotta. Questo schema si chiama **avvicendamento dei processi in memoria (SWAPPING)**

## 

## Cambio di contesto

A volte sono le interruzioni ad indurre il SO a sospendere un processo ed eseguire routine del kernel. In presenza di una interruzione, il sistema deve salvare il **contesto** del processo corrente, per poterlo poi ripristinare quando il processo stesso potrà ritornare in esecuzione. Il contesto e rappresentato all’interno del PCB del processo.

Il passaggio della CPU a un nuovo processo implica la registrazione dello stato del processo vecchio e il caricamento dello stato precedentemente registrato del nuovo processo.

Questa procedura e nota col nome di **cambio di contesto** *(context switch*).

## 3.OPERAZIONE SUI PROCESSI //

## Creazione di un processo

## Terminazione di un processo

## 4.COMUNICAZIONE TRA PROCESSI

I processi eseguiti concorrentemente nel SO possono essere indipendenti o cooperanti.

-Un processo è indipendente se non può influire su altri processi , nè tanto meno può subirne l'influsso ;

-Un processo è cooperante se influenza o può essere influenzato da altri processi.

Un ambiente che consente la cooperazione tra processi può essere utile per diversi ragioni:

* Condivisione dell’informazione, poiché più utenti possono essere interessati alla stessa informazione
* Accelerazione del calcolo, se le attività vengono suddivise ed eseguite parallelamente
* Modularità
* Convenienza

Per lo scambio di informazioni tra processi, vi sono delle tecniche di comunicazione dette IPC. Tali tecniche prevedono diversi metodi per lo scambio di informazioni : attraverso memoria condivisa oppure attraverso scambio di messaggi. Lo scambio di messaggi è utile quando si devono trasmettere piccole quantità di informazioni, altrimenti è meglio usare la memoria condivisa.

## Sistemi a memoria condivisa

La comunicazione tra i processi avviene sulla base di una memoria condivisa che attraverso opportune chiamate di sistema tale memoria viene collocata nello spazio degli indirizzi del processo e successivamente si può accedere a tale memoria attraverso il suo identificatore univoco. Altri processi che intendono usarla devono annetterla al proprio spazio degli indirizzi. La memoria condivisa può essere illustrata dal problema del produttore/consumatore. Nella memoria condivisa vengono tenuti gli elementi prodotti dal produttore e consumati dal consumatore. Vi sono 2 tipi di buffer: buffer illimitato che non pone limiti mentre il buffer limitato pone una dimensione al numero massimo di elementi che si possono produrre.

## Sistemi a scambio di messaggi

Un altro modo per la comunicazione tra processi è quello dello scambio di informazioni attraverso lo scambio dei messaggi. La comunicazione avviene senza la condivisione dello spazio degli indirizzi. Avviene tramite le chiamate di sistema send e receive. Vi possono essere diversi tipi di comunicazione:

-comunicazione diretta o indiretta ;

-comunicazione sincrona o asincrona ;

-gestione automatica o esplicita del buffer;

Per comunicare i processi devono disporre della possibilità di far riferimento ad altri processi.

NOMINAZIONE

Con la comunicazione diretta ogni processo che invia o riceve dati, oltre al messaggio deve specificare il processo con cui vuole comunicare. Questo tipo di comunicazione ha la seguente caratteristica: ogni processo deve chiamarsi a vicenda e viene instaurato un canale tra i 2 processi. Questo tipo di scambio è detto simmetrico ; una variante asimmetrica avviene tramite la specificazione del processo ricevente all'interno del processo mittente.Il processo ricevente può ricevere un messaggio da qualsiasi processo che lo invoca.

Con la comunicazione indiretta, lo scambio dei messaggi avviene tramite delle porte che sono degli oggetti in cui i processi prendono e mettono messaggi. La stessa porta deve essere condivisa da i processi che intendono usare questo tipo di comunicazione.

SINCRONIZZAZIONE

La comunicazione tra processi avviene attraverso chiamate delle primitive send e receive.

-Invio sincrono : il processo che invia il messaggio si blocca finchè il ricevente riceva il messaggio(può essere una porta o un processo);

-Invio asincrono: il processo invia il messaggio e continua la sua esecuzione ;

-Ricezione sincrona: il ricevente si blocca nell'attesa di ricevere un messaggio ;

-Ricezione asincrona: il ricevente riceve un messaggio valido o un valore nullo ;

CODE DI MESSAGGI

## Se la comunicazione è diretta o indiretta , i messaggi scambiati tra processi comunicanti risiedono in code temporanee. Fondamentalmente esistono 3 modi per realizzare queste code :

-Capacità 0 : lunghezza max = 0 . In questo caso il trasmittente deve fermarsi finchè il ricevente non riceva il messaggio ;

-Capacità limitata: può al massimo contenere n messaggi.Se il canale è pieno il trasmittente deve fermarsi in attesa che ci sia spazio disponibile nella coda ;

-Capacità illimitata : lunghezza indefinita , il trasmittente non si ferma mai.

Il caso con capacità 0 è detto sistema a scambio di messaggi NO BUFFERING(senza memorizzazione transitoria); gli altri 2 sistemi AUTOMATIC BUFFERING(memorizzazione transitoria automatica).

## 5.COMUNICAZIONE CLIENT SERVER (facoltativo)

Una socket è definita come l’estremità di un canale di comunicazione. Una coppia di processi che comunica attraverso una rete usa una coppia di socket, una per ogni processo e ogni socket è identificata attraverso un indirizzo IP concatenato a un numero di porta. Il cliente comunica con il server creando una socket e collegandosi per suo tramite alla porta su cui il server è in ascolto. Stabilita la connessione, il client può leggere dalla socket tramite le ordinarie istruzioni di IO. I messaggi vengono suddivisi in pacchetti ordinati correttamente ed inviati sulla socket. Si indirizzano ad un demone il quale provvederà a comporre il messaggio originale.