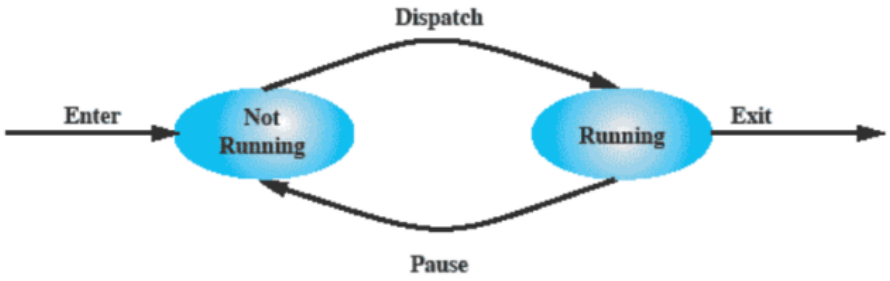
Modelli

Index

Modello dei processi a 2 stati

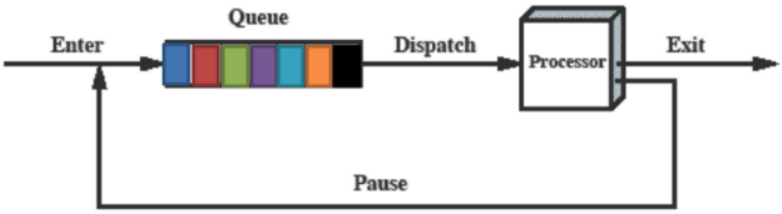
Un processo potrebbe essere in uno di questi due stati (non consideriamo infatti creazione e terminazione)

- in esecuzione
- non in esecuzione (anche quando viene messo in pausa dal dispatcher)



Dal punto di vista dell'implementazione avremmo una coda in cui sono processi tutti i processi che non sono in esecuzione, il dispatch quindi prende il processo in cima alla queue (quello in nero) e lo mette in esecuzione.
I processi vengono quindi mossi dal dispacher dalla CPU alla coda e viceversa, finché

il processo non viene completato



Creazione e terminazione di processi

Creazione

In ogni istante in un sistema operativo sono $n \geq 1$ processi in esecuzione (come minimo ci sta un'interfaccia grafica o testuale in attesa di input). Quando viene dato un comando dall'utente, quasi sempre si crea un nuovo processo

process spawning → è un fenomeno che si verifica quando un processo in esecuzione crea un nuovo processo. Si hanno quindi un **processo padre** (quello che crea) e un **processo figlio** (processo creato) passando quindi da n a $n + 1$ processi

Terminazione

Con la terminazione si passa da $n \geq 2$ processi a $n - 1$. Esiste in oltre sempre un processo "*master*" che non può essere terminato (salvo spegnimento del computer)

Normale completamento

Il normale completamento di un processo (come precedentemente nominato) avviene con l'istruzione macchina `HALT` che genera un'interruzione (che nei linguaggi di alto livello è invocata da una system call, inserita automaticamente dai compilatori dopo l'ultima istruzione di un programma)

Uccisioni

Oltre al normale completamento ci stanno le uccisioni, eseguite generalmente dal SO a causa di errori come ad esempio:

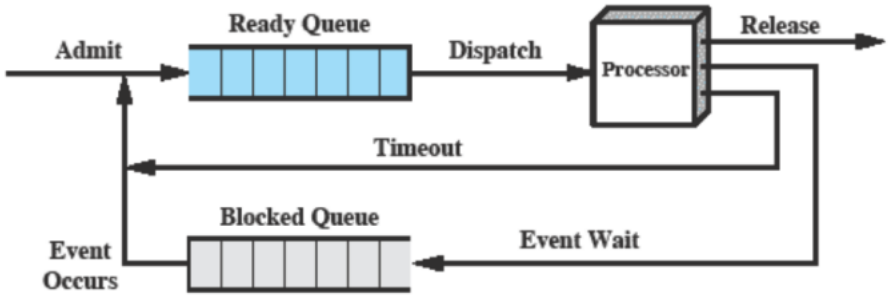
- memoria non disponibile
- errori di protezione
- errore fatale a livello di istruzione (divisione per zero ecc.)
- operazione di I/O fallita
  Oppure dall'utente (es. X sulla finestra) o da un altro processo (es. invio segnale

da terminale)

Modello dei processi a 5 stati

In questo caso il momento in cui il processo non è in esecuzione è diviso tra quando il processo è ready e quando il processo è blocked

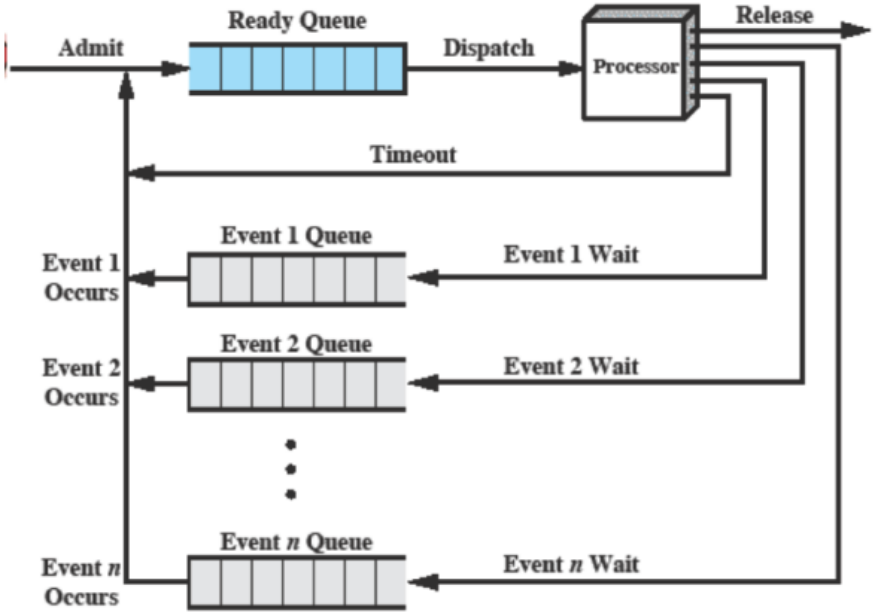
Una volta creato un processo diventa subito ready, può diventare running attraverso il dispatch. Se sono running e ad un certo punto nelle istruzioni eseguo un'operazione I/O entro in attesa (blocked) e ci rimango finché il dispositivo I/O non ha terminato e a quel punto ritorno ready



Avendo due stati necessiterà quindi di due code (al posto di una sola). Si aggiunge infatti la coda di blocked.

Assumiamo che tutti questi processi siano in RAM, ci sono svariati motivi per cui un processo possa esser messo in attesa, infatti i sistemi operativi non hanno una coda per tutti gli eventi, ma ne hanno una per ogni evento (per ogni motivo per cui il

processo è stato messo in attesa)



Processi sospesi

Potrebbe succedere che molti processi possano essere in attesa di input/output. Finché sono blocked questi stanno solamente occupando inutilmente della memoria RAM. Dunque quello che viene fatto è spostare (*swappare*) alcuni processi sul disco e, quando dovranno essere ripresi, vengono rispostati sulla RAM.
Dunque lo stato “blocked” diventa “suspendend” quando viene swappato su disco

Questo porta all’introduzione di due nuovi stati:

- *blocked/suspended* → swappato mentre era bloccato
- *ready/suspended* → swappato mentre non era bloccato

Abbiamo quindi un totale di 7 stati:

Adesso quindi una volta creato il processo il SO decide se metterlo in RAM (Ready) oppure swapparlo su disco (Ready/Suspend). Il dispatch sceglie solo tra i processi in RAM. Quando il processo fa qualche richiesta bloccante, diventa blocked, da cui si sblocca solo quando la richiesta soddisfatta. Un processo blocked può essere swappato su disco. Da Running anche può essere swappato

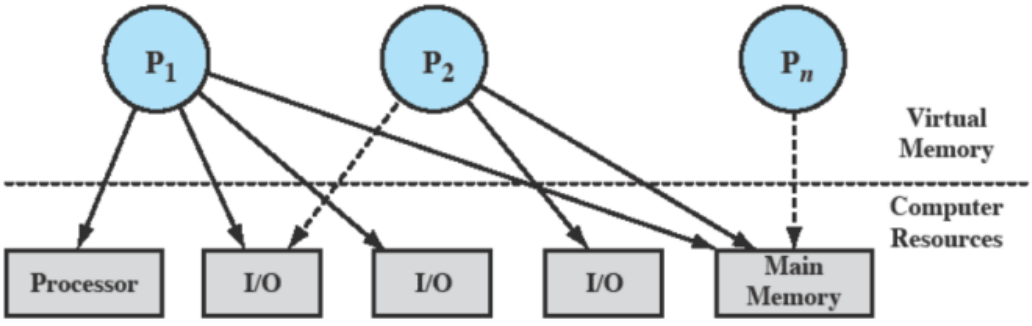
Motivi per sospendere un processo

Motivo	Commento
Swapping	Il SO ha bisogno di eseguire un processo con alta priorità (o è molto grande) e per fargli spazio bisogna swappare dei processi ready
Interno al SO	Il SO sospetta che il processo stia causando problemi
Richiesta utente interattiva	Ad esempio: debugging
Periodicità	Il processo viene eseguito periodicamente e può venire sospeso in attesa della prossima esecuzione
Richiesta del padre	Il padre potrebbe voler sospendere l'esecuzione di un figlio per esaminarlo o modificalo per coordinare l'attività tra più figli

---

Processi e Risorse

Il sistema operativo oltre a dover gestire i processi deve anche gestire come questi richiedono e acquisiscano delle risorse (tipicamente dispositivi di input).



Con le linee piene si intendono risorse che sono state acquisite dai processi, mentre con le linee tratteggiate delle risorse che sono state solamente richieste dai processi

(ma non concesse, ad esempio perché è una risorsa esclusiva, ovvero fin tanto che lo ha P_1 non lo può avere P_2)