Vita di un processo

Modalità di esecuzione

La maggior parte dei processori supporta almeno due modalità di esecuzione (il Pentinum ne ha 4):

- modo sistema (*kernel mode*) → in cui si ha il pieno controllo e si può accedere a qualsiasi locazione di memoria (compresa quella del kernel)
- modo utente → molte operazioni sono vietate

Kernel Mode

La **kernel mode** serve per le operazioni eseguite dal kernel. Viene utilizzata per:

- gestione dei processi (tramite PCB)
  - creazione e terminazione
  - pianificazione di lungo, medio e breve termine (*scheduling* e *dispatching*)
  - avvicendamento (*process switching*)
  - sincronizzazione e comunicazione
- gestione della memoria principale
  - allocazione di spazio per i processi
  - gestione della memoria virtuale
- gestione dell'I/O
  - gestione dei buffer e delle cache per l'I/O
  - assegnazione risorse I/O ai processi
- funzioni di supporto (gestione interrupt, accounting, monitoraggio)

Da User Mode a Kernel Mode e ritorno

Si basa su un'idea semplice: un processo utente inizia sempre in modalità utente, ma cambia e si porta in modalità sistema in seguito ad un interrupt (come abbiamo visto infatti una volta eseguita un'interrupt viene eseguita una parte di codice di sistema). La prima cosa che fa l'hardware, prima di cominciare alla procedura di sistema da eseguire, cambia modalità passando in kernel mode; questo permette di eseguire l'interrupt handler in kernel mode.

L'ultima istruzione dell'interrupt handler, prima di ritornare al processo di partenza, ripristina la modalità utente

Dunque un processo utente può cambiare modalità a sé stesso, ma **solo per eseguire software di sistema**.
Si ha quindi questo cambiamento in seguito a (esplicitamente voluti):

- system call
- in risposta ad una sua precedente richiesta di I/O (in generale di risorse)

Codice eseguito per conto dello stesso processo interrotto, che non lo ha esplicitamente voluto:

- errore fatale (*abort*) → il processo spesso viene terminato
- errore non fatale (*fault*) → viene eseguito un qualcosa prima di tornare in user mode e continuare il processo

Codice eseguito per conto di qualche processo

- In particolare avviene quando un processo A ha fatto una richiesta I/O quindi il SO lo ho messo in blocked, il SO intanto mette in esecuzione un secondo processo B, nel mentre viene esaudita la richiesta di A ma ciò avviene per conto del processo B

System call sui Pentium

Il codice per una system call sui Pentium è strutturata così:

1. si preparano gli argomenti della chiamata mettendoli in opportuni registri
   tra di essi ci sta il numero che identifica la system call
2. esegue l'istruzione `int 0x80`, che appunto solleva un interrupt (in realtà un'eccezione)
3. in alternativa, dal Pentium 2 in poi, può eseguire l'istruzione `sysenter`, che omette alcuni controlli inutili

Da notare che anche creare un nuovo processo è una system call: in Linux *fork* (oppure *clone* più generale)

Creazione di un processo

Per creare un processo il sistema operativo deve:

1. Assegnargli un PID unico
2. Allocargli spazio in memoria principale
3. Inizializzare il process control block
4. Inserire il processo nella giusta coda (es. ready oppure ready/suspended)
5. Creare o espandere altre strutture dati (es. quelle per l'accounting)

Switching tra processi

Lo **switching tra processi** consiste nel concedere il processore ad un altro processo per qualche motivo scaturito da un evento ed è l'operazione più delicata quando si tratta di scrivere un sistema operativo in quanto pone svariati problemi tra cui:

- Quali eventi determinano uno switch?
- Cosa deve fare il SO per tenere aggiornate le strutture dati in seguito ad uno switch tra processi?

> ⚠ **Attenzione a distinguere lo switch di modalità (da utente a sistema) e lo switching di processi**

Quando effettuare uno switch?

Uno switch può avvenire per i seguenti motivi

Meccanismo	Causa	Uso
Interruzione	Esterna all'esecuzione dell'istruzione corrente	Reazione ad un evento esterno asincrono; include i quanti di tempo per lo scheduler (per evitare che il processore venga monopolizzato)
Eccezione	Associata all'esecuzione dell'istruzione corrente	Gestione di un errore sincrono
Chiamata al SO	Richiesta esplicita	Chiamata a funzione di sistema (caso particolare di eccezione)

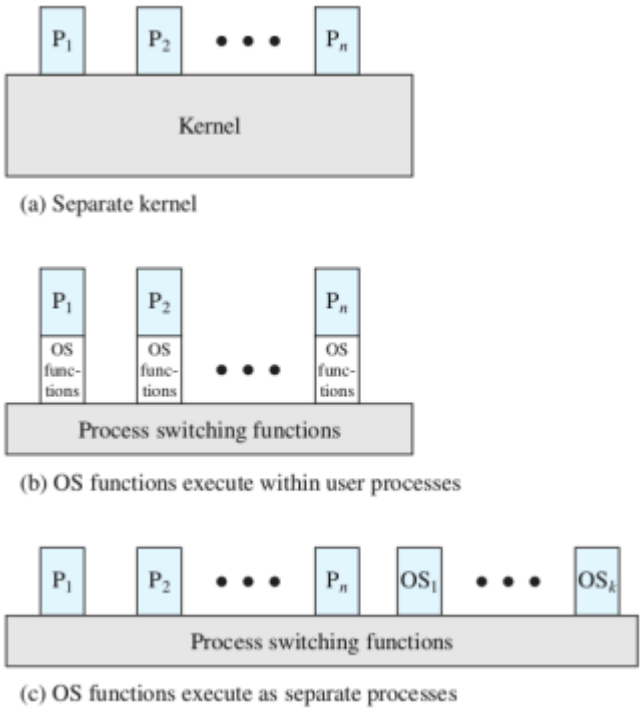
Passaggi per lo switch

Quando si deve sostituire un processo (*process switch*) per prima cosa si switcha in kernel mode poi:

1. Si salva il contesto del programma (registri e PC salvati nel PCB di quel processo)
2. Aggiornare il process control block per quanto riguarda lo stato, attualmente in running
3. Spostare il process control block nella coda appropriata: ready, blocked, ready/suspended
4. Scegliere un altro processo da eseguire
5. Aggiornare lo stato del process control block del processo selezionato
6. Aggiornare le strutture dati per la gestione della memoria
7. Ripristinare il contesto del processo selezionato

Il SO è un processo?

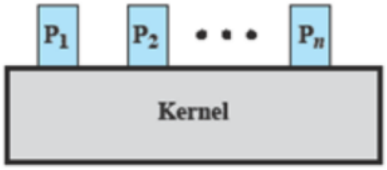
Il SO è solo un insieme di programmi (la maggior parte di questi sono cose che avvengono in seguito ad un interrupt handler) eseguiti sul processore. Semplicemente lascia che altri programmi vadano in esecuzione, per poi riprendere il controllo tramite interrupt



Queste sono le tre possibili configurazioni del SO

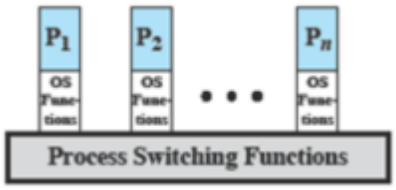
Il Kernel non è un processo

Il Kernel in questo caso si trova al di fuori dei processi lasciando il concetto di processo applicato solo ai programmi utente. Dunque il SO è eseguito come un'entità separata con privilegi più elevati. Ha inoltre una zona di memoria dedicata sia per i dati che per il codice sorgente che per lo stack



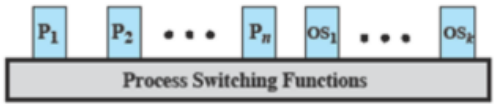
Esecuzione all'interno dei processi utente

Il SO viene eseguito nel contesto di un processo utente (cambia solo la modalità di esecuzione). Non c'è bisogno di un process switch per eseguire una funzione del SO, solo del mode switch. Comunque lo stack delle chiamate rimane separato tra utente e SO. Il process switch è presente, solo eventualmente, alla fine, se lo scheduler decide che tocca ad un altro processo.



SO è basato sui processi

In questo caso tutto è considerato un processo (comprese le funzioni di sistema) fatta eccezione per le funzioni per fare process switching. Il SO viene implementato infatti come un insieme di processi di sistema (ovviamente con privilegi più alti) e partecipano alla competizione per il processore accanto ai processi utente. Alcuni sistemi operativi preferiscono questa modalità in quando più modulare seppur meno efficiente della precedente



Caso concreto: Linux

Qui si ha una via di mezzo tra la seconda e la terza opzioni. Le funzioni del kernel infatti sono per lo più eseguire tramite, per conto del processo corrente (può succedere quindi, per interrupt asincroni, che la gestione di un interrupt causato da un certo processo sia effettuata durante l'esecuzione di un altro processo).
Ci sono però anche dei processi di sistema (*kernel threads*), creati in fase di inizializzazione, che partecipano alla normale competizione del processore, senza essere invocati esplicitamente:

- operazioni periodiche → creare spazio usabile nella memoria principale liberando zone non usate
- scrivere sui dispositivi I/O le operazioni bufferizzare in precedenza
- eseguire operazioni di rete