Processi

Programma in esecuzione, detto terra terra.

Ogni sistema operativo esegue più processi allo stesso tempo, gestore di processi

Un processo si evolve e passa attraverso vari stati → fotografia che si può fare in quel momento del processo → insieme dei registri del processore e il set di variabili che appartengono al quel processo → fotografia delle variabili che usa

Stati

1. Starting
2. Ready
3. Executing
4. Blocked - aspettando la condizione, il task si deschedula e riprende solo quando si arriva alla condizione
5. Terminating

`a` → Creation

`b1` → Scheduling: decide quale sarà il prossimo processo ad andare in esecuzione

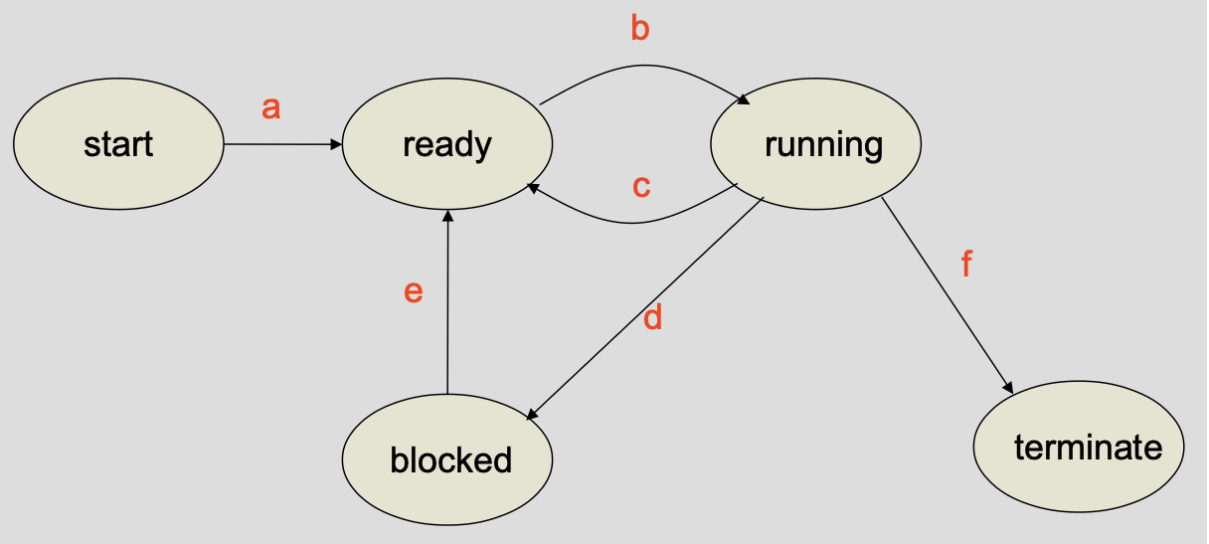
`b2` → Dispatch: colui che mette il processo in esecuzione

`c` → Preemption: deschedulazione prima di aver finito quando arriva una task a più alta priorità al posto mio, prima salvo lo stato/i registri (**CONTESTO**), così non appena torna running non devo ricominciare da capo

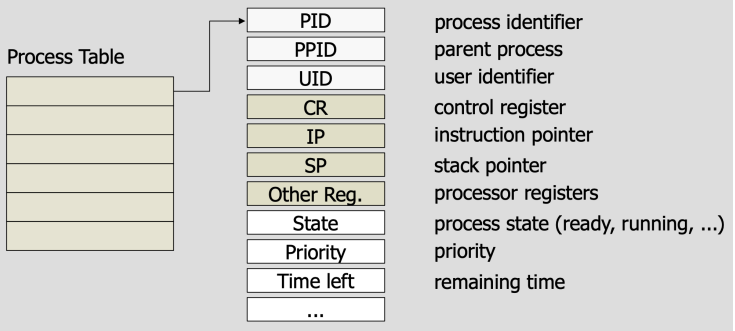
`d` → Wait on condition: semafori, mutex, variabili di condizione

`e` → Condition true

`f` → Exit



Process Control Block

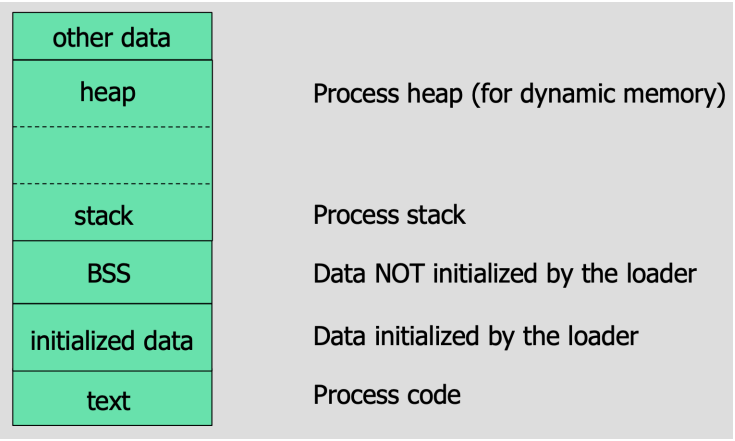


Ogni processo ha un PCB → serie di info, identificatori, puntatori che identificano un determinato processo.

Sono presenti un tot di registri collegati al processo (i registri stanno nel processore) → servono quando c'è un **CONTENT SWITCH (ripristino dello stato del processo dopo una wait).**

Può essere acceduto dal SO (Scheduler, Virtual Memory, ecc), mentre l'utente può accedere solo a certe info del PCB.

Memory Layout Process



Stack → luogo delle automatic variables, allocate durante l'esecuzione di una chiamata funzione o di un programma. Area di memoria dove sono memorizzate lo stato di esecuzione del programma, gli alias delle variabili locali di ogni metodo, il loro indirizzo di memoria e il loro valore.

Heap → spazio dedicato alla memoria dinamica dove sono memorizzati gli oggetti e il loro stato.

Text → codice del processo

Other Data → PCB in Linux

Le cose funzionano bene se un processo vive nel suo mondo e non rompe le scatole allo spazio di indirizzamento di altri processi, non condividono memoria! → **SEGMENTATION FAULT.**

Al massimo possono condividere la zona text mandando un puntatore.

Modalità di operazione

- User → user code
- Supervisor → OS

Per capire in quale modalità il processo sta andando basta guardare il bit Control Register CR.

Quando cambio modalità si dice **MODE SWITCH (**Possono avvenire in caso di Interrupts o Traps**)**

Process Switch o Context Switch

Operazione time consuming.

Avviene o in caso di Preemption (esistono schede non preemptive → anche se arriva un processo ad alta priorità, aspettano che quello in esecuzione finisca per risparmiare risorse) o quando un processo di blocca o quando nei sistemi time-sharing un processo ha finito il suo tempo di esecuzione.

Exec pointer

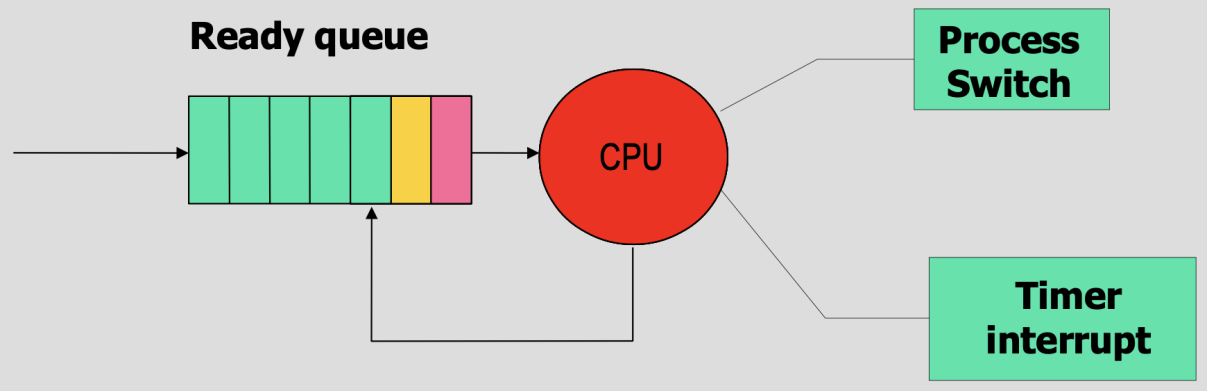
Puntatore del SO, al PCB del processo attualmente RUNNING in esecuzione. C'è un exec pointer per ogni processore.

---

1. Salvo i parametri nel mio stack
2. Se devo entrare in ambito privilegiato cambio a supervisor mode
3. Salvo il contesto nel PCB di colui che sta eseguendo. In exec trovo il PCB del processo che sta eseguendo, lì devo salvare il suo contesto incluso lo STACK POINTER.
4. Chiamo lo scheduler che si trova nel supervised space
    a. Il processo che era exec viene messo nella coda dei processi bloccati.
    b. Seleziona un altro processo ad andare RUNNING.
    c. Sarà il suo PCB ad essere puntato dall'exec pointer.
5. Restore del processo exec
6. Restore del contesto, incluso lo STACK POINTER

Time Sharing Systems

Ogni processo è eseguito per massimo un intervallo di tempo. Alla fine del tempo il processore viene assegnato a qualcun'altro.



Cause del Context Switch

- Volontarie → il processo chiama una primitiva bloccante, tipo semafori.
- Non volontario → triggherato da un interrupt.

Processi

- Proprietà delle risorse → un processo è proprietario di uno spazio di indirizzamento, o di canali di I/O.
- Flusso di esecuzione e scheduling → si segue una specifica traccia attraverso una sequenza di stati interni.

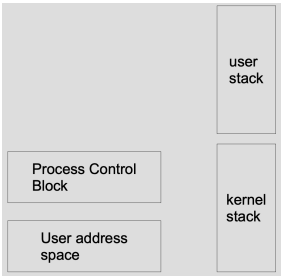
Multi-Threading

Il processo pesante è il proprietario delle risorse, dello spazio di indirizzamento e dei canall di I/O.

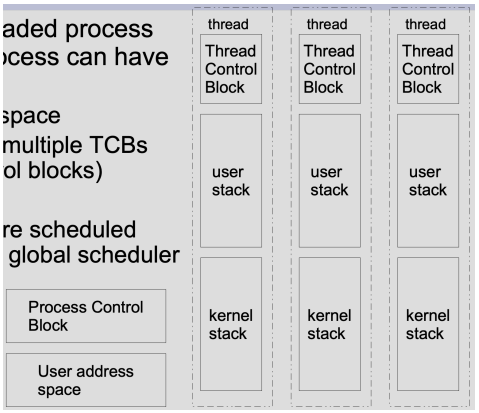
Thread → processo leggero che si riferisce al flusso di esecuzione e scheduling, entità di scheduling indipendenti che concorrono insieme sfruttando le risorse di proprietà del loro processo.

Distinguere vari flussi di esecuzione all'interno di una stessa proprietà.

Un processo può consistere in più thread, ma non il contrario.



Single-Thread



Multi-Thread

In generale i processi non condividono memoria, ma per comunicare c'è bisogno di effettuare delle chiamate a primitive di sistema, CONTEXT SWITCH.

La comunicazione tra thread e il THREAD SWITCH sono più semplici perché lo spazio di indirizzamento è sempre lo stesso.