Laboratorio di Reti

Federico Matteoni

1 Threads

Processo Istanza di un programma in esecuzione

Thread **Flusso di esecuzione** all'interno di un processo ⇒ Ogni processo ha almeno un thread.
I **thread condividono le risorse di un processo**.
Possono essere eseguiti sia su single-core (es. interleaving, time-sharing...) che su multicore (più flussi di esecuzione in parallelo)

Multitasking Si può riferire a

Processi, controllato esclusivamente dal S.O.

Thread, controllato in parte dal programmatore

Contesto di un processo Insieme delle informazioni necessarie per ristabilire esattamente lo stato in cui si trova il S.O. nel momento in cui si interrompe l'esecuzione di un processo per passare ad un altro: registri del processore, memoria del processo...

Perché? Per gestire più funzionalità contemporaneamente, come gestire input, visualizzare a schermo, monitorare la rete ed eseguire calcoli.
Esempi noti: browser web, videogame multiplayer. Si creano **più componenti interagenti** in modo da:

Usare meglio le risorse

Migliorare le performance per applicazioni che richiedono grossi calcoli: si dividono i task per eseguirli in parallelo.

Anche problemi: difficile debugging e manutenzione, sincronizzazione, deadlocks...

In Java Il main thread, invocato dalla JVM all'esecuzione del programma, può attivare altri thread. La JVM attiva automaticamente altri thread come il garbage collector.
Un thread è un oggetto. Per creare un thread si definisce un task che implementi l'interfaccia `Runnable` e si crea un thread passandogli l'istanza del task creato. Altrimenti si può estendere la classe `java.lang.Thread`.

Runnable Appartiene a `java.language`, contiene solo la firma del metodo `void run()`. Un oggetto che la implementa è un frammento di codice che può essere eseguito in un thread.

Stati

Created/New: subito dopo l'istruzione `new`, variabili allocate ed inizializzate. Thread in attesa di passare in esecuzione

Runnable/Running: thread in esecuzione o in attesa per ottenere la CPU (Java non separa i due stati).

Not Runnable (**Blocked/Waiting**): thread non può essere messo in esecuzione, può accadere quando attende un'operazione I/O o ha invocato metodi come `sleep()` oppure `wait()`.

Dead: termine naturale o dopo l'invocazione di `stop()` da parte di altri thread (deprecato).