Capitolo 4: Thread

- Introduzione
- Programmazione multicore
- Modelli di supporto al multithreading
- Librerie dei thread
- Threading implicito
- Problematiche di programmazione multithread
- Esempi di Sistemi Operativi

Thread

- In alcune situazioni una singola applicazione deve poter gestire molti compiti simili tra loro.
- In altre una singola applicazione può dover gestire più compiti diversi, a volte eseguibili concorrentemente.
- Una possibile soluzione è quella della creazione di più processi tradizionali.
- Nel modello a processi, l'attivazione di un processo o il cambio di contesto sono operazioni molto complesse che richiedono ingenti quantità di tempo per essere portate a termine.
- Tuttavia a volte l'attività richiesta ha vita relativamente breve rispetto a questi tempi.
  - Ad es. l'invio di una pagina html da parte di un server web: applicazione troppo leggera per motivare un nuovo processo.
- Possibile soluzione: threads.

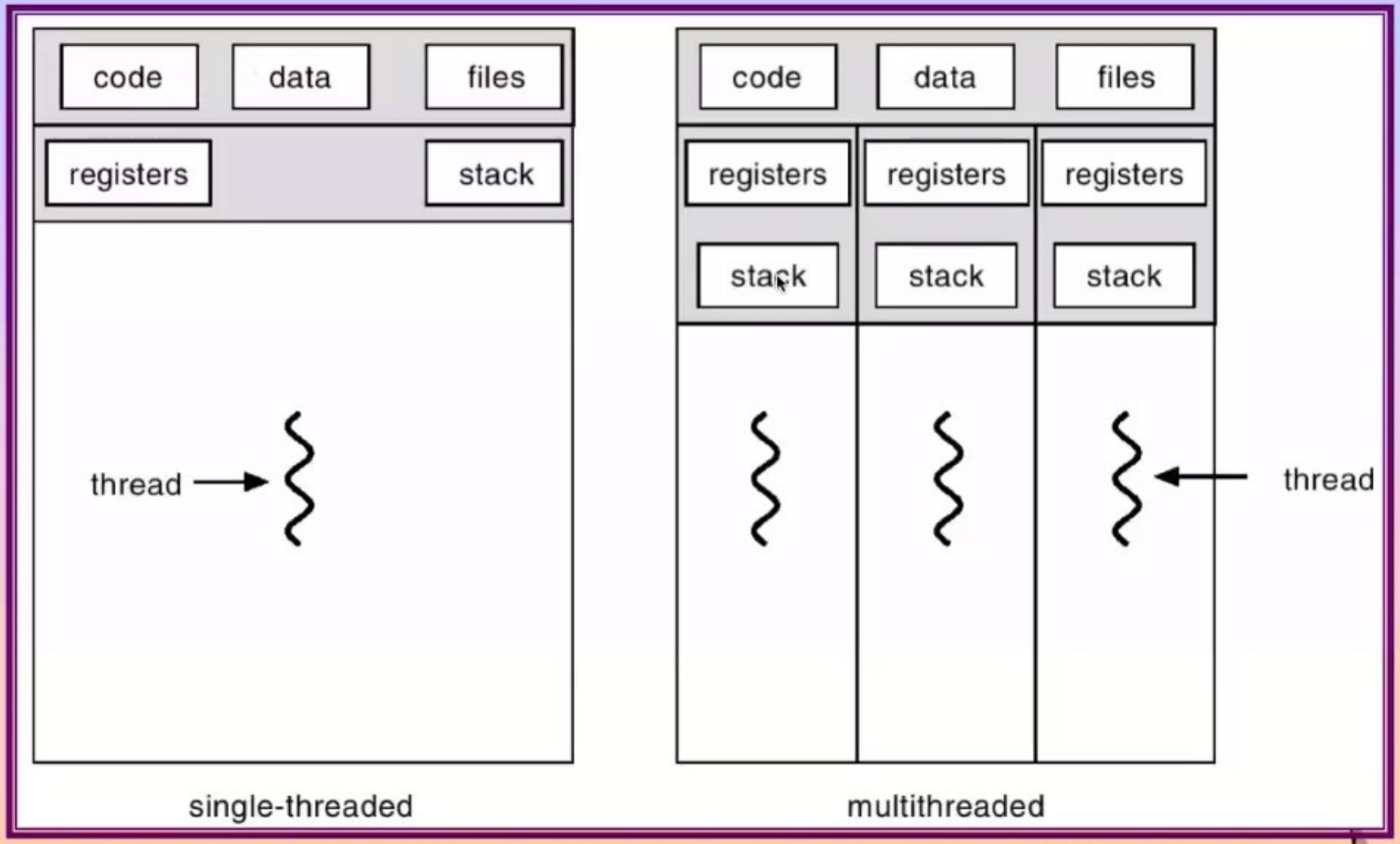
Thread (II)

- Un thread è talvolta chiamato processo leggero (*lightweight process*).
- Condivide con gli altri thread che appartengono allo stesso processo la sezione del codice, la sezione dei dati e altre risorse di sistema, come i file aperti e i segnali.
- Processi tradizionali = singolo thread.
- Processi multithread = più thread.
- Molti kernel sono ormai multithread,
  - con i singoli thread dedicati a servizi specifici come la gestione dei dispositivi periferici o delle interruzioni.
- Molti programmi per i moderni PC sono predisposti per essere eseguiti da processi multithread.
- Un'applicazione, solitamente, è codificata come un processo a sè stante comprendente più thread di controllo.
- Il multithreading è la capacità di un sistema operativo di supportare thread di esecuzione multipli per ogni processo.

Il modello a thread

- Le idee di base dietro il modello a thread sono:
  - Permettere la definizione di attività “leggere” (ligthweight processes--LWP) con costo di attivazione e terminazione limitato.
  - Possibilità di condividere lo stesso spazio di indirizzamento.
- Ogni processo racchiude più flussi di controllo (thread) che condividono le aree di testo e dei dati.
- Un thread è l’unità di base d’uso della CPU e comprende:
  - Identificatore di thread.
  - Program counter.
  - Insieme di registri.
  - Stack.
- Condivide con gli altri thread che appartengono allo stesso processo:
  - Sezione codice.
  - Sezione dati.
  - Risorse di sistema.
    - ✓ Ad es.: file aperti.

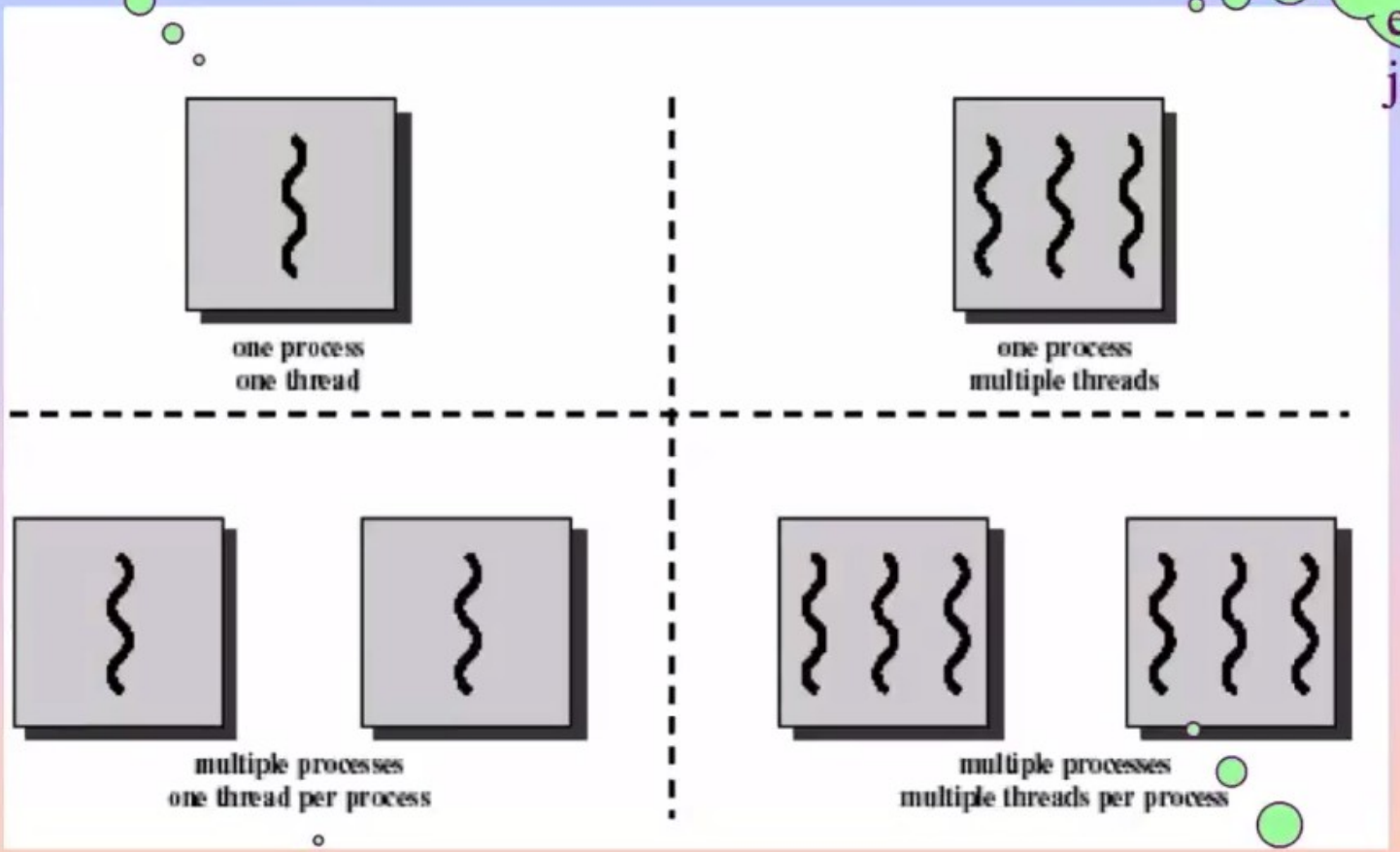
Processi a singolo thread e multithread



Thread e Processi: quattro possibili scenari

MS-DOS

Motore Runtime java

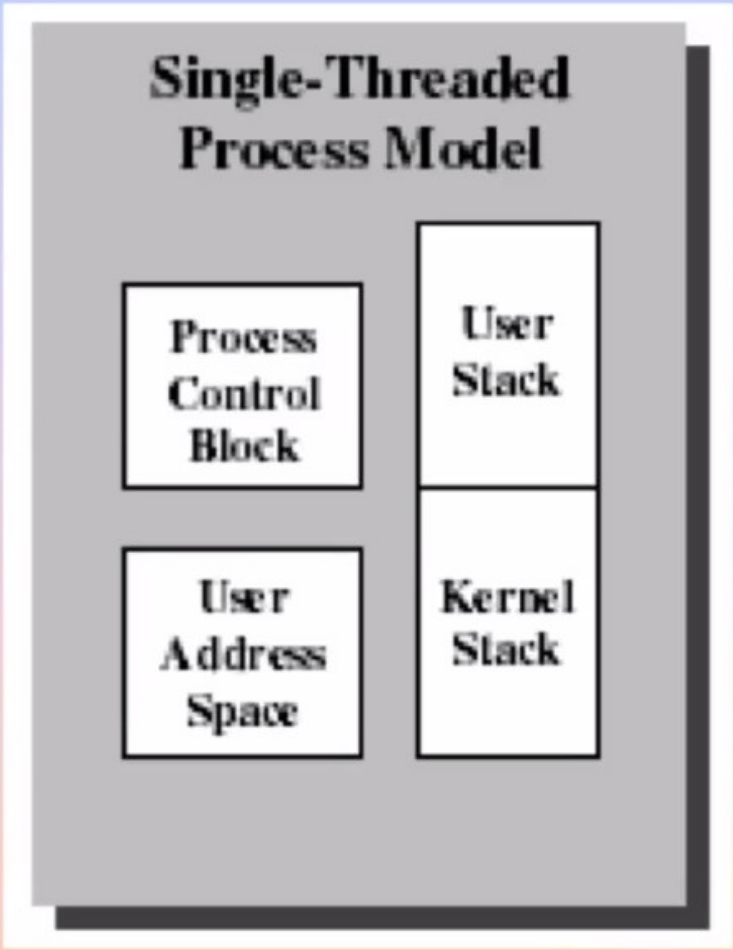


UNIX

WindowsNT
Solaris
Mach

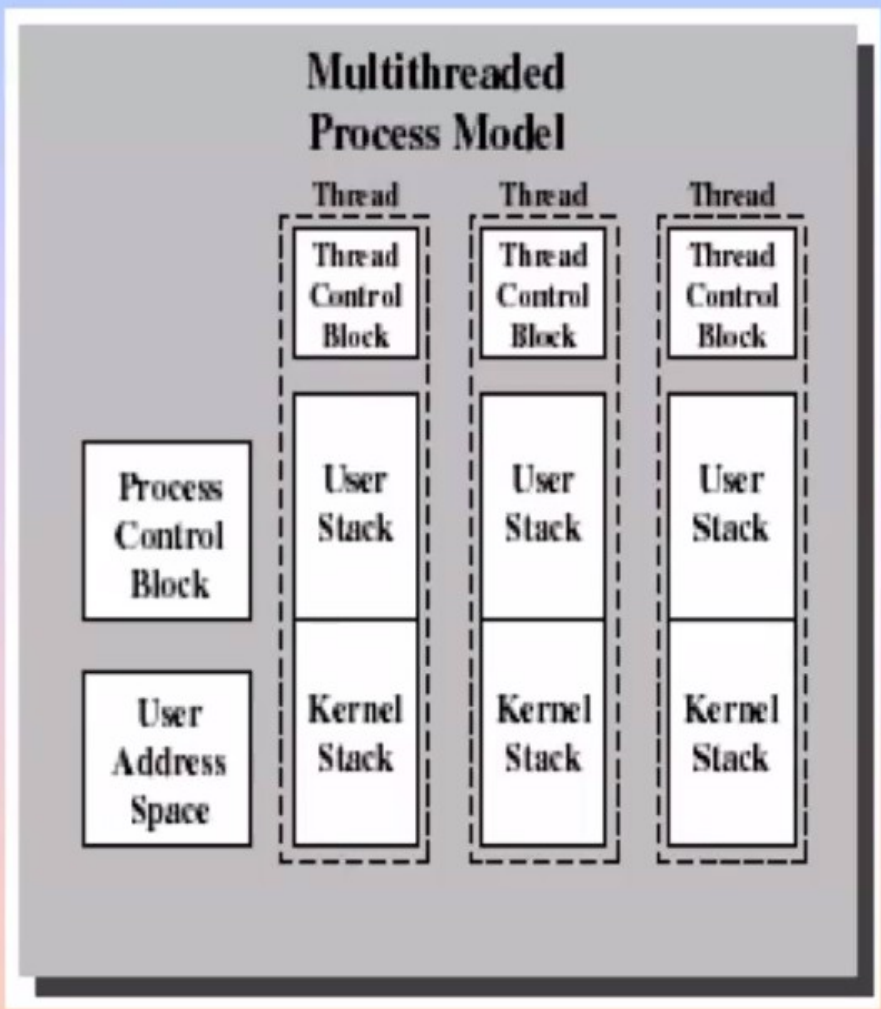
Modello dei processi a thread singolo

- In un modello a thread singolo la rappresentazione di un processo contiene il suo PCB e il suo spazio indirizzamento utente, lo stack utente e lo stack del kernel



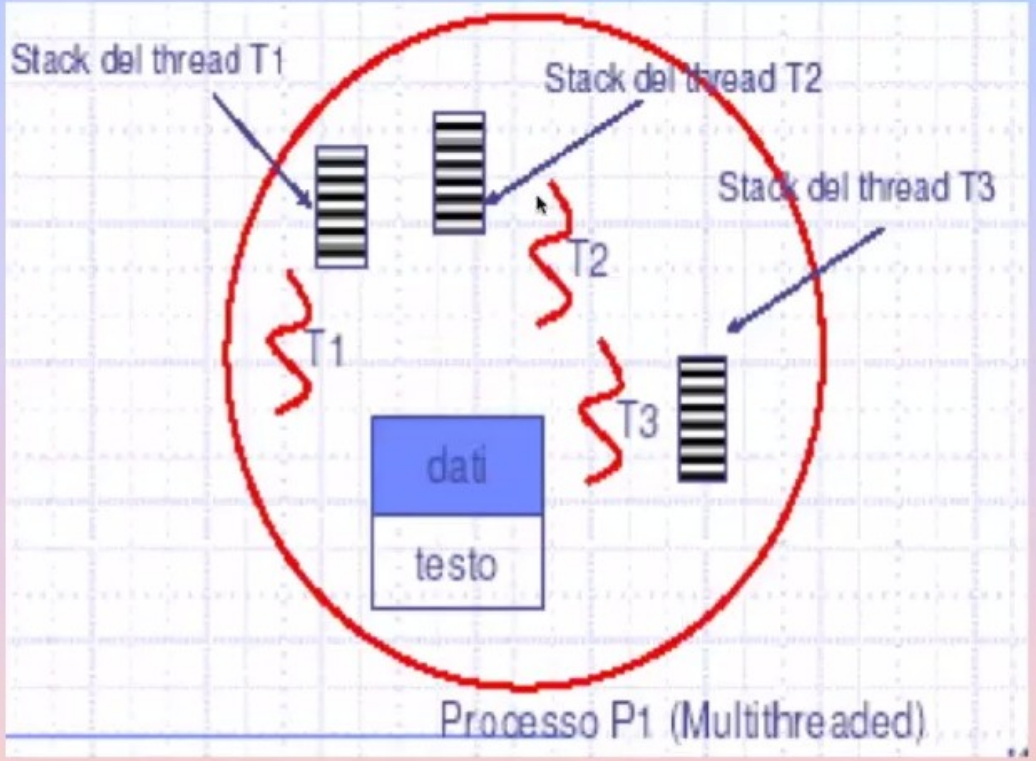
Modello multithread dei processi

- In un ambiente multithreading ogni processo ha associato:
  - un solo PCB
  - un solo spazio di indirizzamento utente
- …ma ogni thread ha un proprio
  - stack
  - un blocco di controllo privato contenente l'immagine dei registri
  - Una priorità
  - Altre informazioni relative al thread



Esempio: struttura processo multithread

- Tutti i thread componenti un processo:
  - condividono lo stato e le risorse del processo
  - risiedono nello spazio di indirizzamento
  - hanno accesso agli stessi dati



- **T1** legge i caratteri da tastiera e li visualizza
- **T2** formatta il testo
- **T3** memorizza periodicamente sul disco

Vantaggi

- La programmazione multithread offre numerosi vantaggi classificabili rispetto a 4 fattori principali:
  - *tempo di risposta*,
  - *condivisione delle risorse*,
  - *economia*,
  - *uso di più unità di elaborazione.*
- **Tempo di risposta:**
  - l'esecuzione può continuare anche se parte del programma è bloccata o stà eseguendo un'operazione particolarmente lunga.
  - Es.: Programma di consultazione web
    - ✓ Un thread carica un'immagine
    - ✓ Un thread permette l'interazione con l'utente

Vantaggi (II)

- **Condivisione delle risorse:**
  - I thread normalmente condividono la memoria e le risorse del processo cui appartengono.
- Possiamo avere quindi più thread di attività diverse nello stesso spazio indirizzi.
- Questa condivisione rende più conveniente creare thread e gestire cambiamenti di contesto piuttosto che creare nuovi processi
- **Economia:**
  - Assegnare memoria e risorse per la creazione di nuovi processi è costoso, cosi' i cambi di contesti.
- Con i thread queste operazioni sono alleggerite.
- **Uso di più unità di elaborazione:**
  - I thread di uno stesso processo possono essere eseguiti in parallelo.

Svantaggi

- Il modello a thread presenta ovviamente anche alcuni svantaggi.
- Maggiore complessità di progettazione e programmazione:
  - I processi devono essere “pensati” paralleli.
  - Difficile sincronizzazione tra i thread.
- Il modello è inadatto per situazioni in cui i dati devono essere protetti.
- La condivisione delle risorse accentua il pericolo di interferenze.