SC - 13/5/2025

lunedì 12 maggio 2025 20:07

THREAD

Ogni processo è costituito da una o più unità di esecuzione dette "thread". Un thread possiamo immaginarlo come una sequenza di istruzioni del programma che può essere eseguita in modo indipendente all'interno di un processo.

Diversamente da quanto visto per i processi, più thread associati allo stesso processo condividono:

- codice
- quasi tutti i dati in memoria (costanti, var. globali, heap)
- file aperti
- segnali e loro routine di gestione
- directory di lavoro corrente
- diritti dell'utente proprietario del processo

TUTTI i thread associati ad uno stesso processo ne condividono l'immagine di memoria.

Ciascun thread gestisce in modo indipendente:

- Un identificativo univoco (thread ID)
- stack
- registri
- errno
- metadati (proprietà specifiche per ogni thread)

Complementarmente:

① Un thread esiste all'interno di un processo e ne usa le risorse.

② Ogni thread ha un controllo del flusso di esecuzione del codice indipendente

③ Un thread duplica le risorse MINIME necessarie per la sua esecuzione

④ Un thread condivide le risorse del processo in cui viene eseguito con gli altri thread associati allo stesso processo.

COME USARE I THREAD

L'interfaccia di programmazione che adottiamo per il multithreading è quella offerta dai "POSIX threads"

IMPORTANTE: compilare con gcc usando "-lpthread"

```
#include <pthread.h>

int pthread_create (pthread * thread,          → In caso di successo, contiene il thread ID
                    const pthread_attr_t * attr,   → Struct per specificare attributi del thread
                    void * (* start_routine) (void *),   → puntatore alla funzione da eseguire
                    void * arg)                → eventuale parametri per la funzione
↑
0 se tutto OK
o numero errore

void pthread_exit (void * retval)   → Termina il thread restituendo retval

int pthread_join (pthread thread_id,   → ID del thread di cui si vuole attendere la conclusione
                  void ** retval)      → puntatore a variabile in cui verrà scritto il valore di ritorno
```

Possiamo attendere con pthread_join solo la terminazione dei thread "joinable" al thread che li ha creati.

```
int pthread_detach (pthread thread_id)   → "distacca" il thread thread_id da quello che lo ha creato
```