Sistemi Operativi - Tutoraggi

Laurea in Ingegneria Informatica
Università Tor Vergata
Tutor: Romolo Marotta
Docente del corso: Francesco Quaglia

Thread

1. pthread_create, pthread_exit, pthread_join
2. CreateThread, ExitThread, GetExitCodeThread

Processi e threads

- Un programma
- I dati su cui opera
- Almeno uno stack
- Dati di contesto
- Rif. risorse hardware
- Identificativi
- Statistiche
- Uno stato

PCB1

Immagine di processo P1

- Un programma
- I dati su cui opera

- Almeno uno stack
- Dati di contesto
- Identificativi
- Statistiche
- Stato

TCB

Thread A

- Almeno uno stack
- Dati di contesto
- Identificativi
- Statistiche
- Stato

TCB

Thread B

- Rif. risorse hardware
- Identificativi
- Statistiche
- Uno stato

PCB1

Immagine di processo P1

Gestione di thread

- La libreria ULT o il sistema operativo offrono dei servizi per:
  - Creare thread
  - Terminare l'esecuzione del thread corrente
  - Aspettare che uno specifico thread termini la sua esecuzione
- Standard POSIX:
  - pthread_create
  - pthread_exit
  - pthread_join
  - E la exit in ambienti multi-threaded?
    - è tipicamente mappata su un'altra system call (e.g., exit_group)
    - La system call exit termina solo il thread corrente
- Win32
  - CreateThread
  - ExitThread

Esempio 1

Esempio 2

Esercizio 1

- Scrivere un programma per Windows/Unix che permette al processo principale P di create un nuovo thread T il cui percorso di esecuzione è associato alla funzione “thread_function”.
- Il processo principale P ed il nuovo thread T dovranno stampare ad output una stringa che li identifichi rispettando l’ordine T→P, senza utilizzare “WaitForSingleObject”/“pthread_join”, ma sfruttando un concetto fondamentale che accomuna tutti i thread di un determinato processo.

Esercizio 2

- Scrivere un programma per Unix che sia in grado di generare due thread T1 e T2, tali per cui:
- T1 chiede all'utente di inserire una messaggio da tastiera
- T2 scrive il messaggio ottenuto dall'utente a schermo
- non devono essere usate variabili globali.

Esercizio 3

- Scrivere un programma per Windows il cui processo principale genera N threads.
- Ogni thread, con indice i, chiede all'utente di inserire un messaggio per il thread con indice i+1.
- Ogni thread, con indice i, come prima cosa stampa il messaggio ricevuto dal thread con indice i-1.
- Le operazioni di lettura del messaggio ricevuto dal thread precedente, stampa del messaggio e richiesta di un nuovo messaggio per il thread successivo, devono essere sequenzializzate.