Esercitazione [09]

Server multi-thread

Riccardo Lazzeretti - lazzeretti@diag.uniroma1.it

Sistemi di Calcolo 2
Programmazione dei Sistemi di Calcolo Multi-Nodo

Corso di Laurea in Ingegneria Informatica e Automatica

Sommario

- Correzione esercitazione precedente
- Server multi-thread
- Esercizio: completare server multi-thread

- Esercitazione pre-esame

Server multi-thread

- Per ogni connessione accettata, viene lanciato un nuovo thread tramite `pthread_create()`
  - Oltre ai parametri application-specific, il nuovo thread avrà bisogno del descrittore della socket relativa alla connessione appena accettata
  - Come dobbiamo gestire le socket? Ragionateci sopra
  - Una volta completata l'elaborazione della connessione, il thread termina
  - Il main thread può voler fare detach dei thread creati
- Minore overhead rispetto al server multi-process
- Gestione più semplice di eventuali strutture dati condivise
- Un crash in un thread causa un crash in tutto il processo!

Server multi-thread

```
while (1) {
      int client = accept(server, ……);
      <gestione errori>

      pthread_t t;
      t_args = ……
      <includere client in t_args>
      pthread_create(&t,NULL,handler,(void*)t_args);
      <gestione errori>
      pthread_detach(t);
}
```

Gestione socket???

Esercizio proposto: EchoServer multi-thread

- Completare il codice dell'EchoServer in modalità multi-thread
- Sorgenti
  - `Makefile`
  - Client: `client.c`
  - Server: `server.c`
    - compilazione: `-DSERVER_MTHREAD` vs `-DSERVER_SERIAL`
- Suggerimento: seguire i blocchi di commenti inseriti nel codice
- Altro suggerimento:
  Per monitorare a runtime il numero di istanze di processi/thread attivi in un certo momento, lanciare da terminale il comando:

```
ps -e -T | head -1 ; ps -e -T | grep multithread
```