

# Programmazione distribuita I

(01NVWOV)

AA 2010-2011, Esercitazione di laboratorio n. 3

## Esercizio 3.1 (server TCP concorrente)

Sviluppare un server TCP (in ascolto sulla porta specificata come primo parametro sulla riga di comando) che accetti richieste di trasferimento file da client ed invii il file richiesto.

Il server deve implementare lo stesso protocollo usato nell'esercizio 2.3 ed essere di tipo concorrente, attivando un massimo di **tre** figli simultanei.

Usando il client sviluppato nell'esercizio 2.4 provare ad attivare quattro client simultaneamente e verificare che solo tre riescono a collegarsi.

Provare a terminare un client brutalmente (battendo ^C nella sua finestra) e verificare che il server (padre) sia ancora attivo ed in grado di rispondere ad altri client.

## Esercizio 3.2 (client TCP con multiplexing)

Modificare il client dell'esercizio 2.3 per gestire un'interfaccia utente interattiva che preveda i seguenti comandi:

GET file (richiede di fare GET del file indicato)

Q (richiede di chiudere il collegamento col server col comando QUIT dopo che un eventuale trasferimento in corso è terminato)

A (richiede di terminare immediatamente il collegamento col server, anche interrompendo un eventuale collegamento in corso)

L'interfaccia utente deve essere sempre attiva e permettere quindi il "type-ahead", ossia fornire input anche se c'è un trasferimento in corso dal server.

## Esercizio 3.3 (server TCP con pre-forking)

Modificare l'esercizio 3.1 per fare in modo che i processi che accettano richieste di trasferimento files dai client siano creati non appena il server viene lanciato (pre-forking), e rimangano in attesa finché un client si connette. Se un client chiude la connessione, il processo che serviva il client deve procedere a servire un nuovo client in attesa, se presente.

Rendere configurabile il numero di processi figli che vengono lanciati all'avvio del server tramite linea di comando, imponendo un massimo di 10 figli.

Verificare il corretto funzionamento lanciando il server con 2 figli e collegandosi con 3 client, da ciascuno dei quali è stata effettuata una richiesta per un file. Chiudere uno dei tre client e verificare che viene immediatamente servito il client in attesa.

Infine, modificare il server in modo che se la connessione con il client rimane inattiva per un certo periodo di tempo, essa viene terminata dal server, in modo da consentire al figlio che gestiva il client considerato inattivo di essere pronto a gestire un eventuale altro client in attesa di connessione.

Verificare che il server sia in grado di gestire anche il caso in cui un client collegato vada in crash (per esempio se chiuso tramite il comando kill), ossia che il processo che serviva il client si accorga di questa condizione e si renda disponibile per un eventuale nuovo client in attesa.