Progetto 2017-2018

Programmazione di Sistema

Outline

- Sviluppo di un sistema client-server per la gestione di file e l'esecuzione di comandi.
- Il client puo' essere:
  - una singola applicazione (che accetta i comandi definiti di seguito)
  - insieme di applicazioni che implementano i diversi comandi.
  - la sintassi potrebbe essere leggermente diversa per i due case
- Il server sarà implementato come un *demone* (programma in esecuzione in background)

Outline

- Il server deve offrire le seguenti funzionalità:
  - copia di file (comando **copia**).
  - spostamento di file (comando **muovi**).
  - cancellazione di file (comando **cancella**).
  - elenco dei file/dir (nome,dimensione, data ultimo accesso, comando **lista**).
  - creazione di una nuova cartella (comando **crea_cartella**).
  - cancellazione di una cartella (solo se la cartella e' vuota, comando **elimina_cartella**)
  - stampa la cartella corrente (comando **cartella**, solo app. singola client**)**
  - cambia cartella corrente (comando **cambia_cartella**, solo app.singola client)

Esempio (app. multiple)

copia *file1.txt* remote_host:*file2.txt*

- Copia (da un client in rete) il file locale *file1.txt* sul server *remote_host* con il nome *file2.txt*

cancella* remote_host:*file2.txt

- Elimina il file remoto *file2.txt* sul server

Esempio (app. singola)

$ <nome app> remote_host

> copia *file1.txt file2.txt*

- Copia (da un client in rete) il file locale *file1.txt* sul server *remote_host* con il nome *file2.txt*

> *cancella file2.txt*

- Elimina il file remoto *file2.txt* sul server

Dettagli

- L'accesso ai file deve essere possibile sia con path relativo che assoluto.
- Il server deve inoltre essere in grado di eseguire altri comandi (il cui elenco è contenuto in un file di configurazione) attraverso il comando **esegui**.
  - Deve essere possibile contatenare in pipe i comandi, ridirigere l'output verso file, etc..
  - Come default, lo stdin e stdout del comando remoto sarà quello di **esegui** sul client

Esempi (app. multiple)

- **esegui server:comando**
  - esegue *comando* sul *server* (stdout sul client)
- **esegui server:”comando1 | comando 2”**
  - esegue la pipe *comando 1* | *comando2* su *server* (stdout sul client)
- **esegui server:”comando > file”**
  - esegue *comando* su *server*, outp. su file remoto *file*
- **esegui server:comando > file**
  - esegue *comando* sul *server*, outp. su file locale *file*

More Details

- La richiesta deve essere eseguita con le credenziali (*uid, gid*) dell'utente che ha richiesto il comando (non di chi ha eseguito il demone)

- Il server deve supportare connessioni multiple concorrenti, attraverso l'uso di processi (o thread)

More Details

- Le richieste sono ricevute sulla porta TCP indicata nel file di configurazione
- La directory iniziale (root) del server è anch’essa indicata nel file di configurazione
- La navigazione fra cartelle deve essere limitata al sottoalbero individuato dalla root directory
- I comandi indicati nel file di configurazione non possono contenere path, ma solo il *basename*

More Details

- L'accesso ai file deve essere controllato attraverso un meccanismo di locking:
  - un solo processo/thread accede in scrittura
  - accesso concorrente in lettura
- Il path al file di configurazione, contenente:
  - porta da utilizzare
  - directory di partenza
  - elenco dei comandi

  verrà indicato sulla command line

Documentazione!

- Al progetto andrà allegato:
  - Il codice, adeguatamente commentato
  - Una relazione che descriva (e motivi):
    - le scelte progettuali
    - I principali moduli del progetto (macrocomponenti) e la loro implementazione
    - la sintassi dei comandi ed i messaggi di errore