Progetto Programmazione di sistema

# Descrizione del protocollo

Il server invia al client una serie di comandi nel formato seguente:

Invia al client la lista dei processi attivi sul server.

* list\n\r numero\_processi lista\_processi

Dove:

* + **numero\_processi:** intero senza segno su 32 bits
  + **lista\_processi:** sequenza di pid dim\_nomeProcesso nome\_processo dim\_icona icona

con:

* **pid:** intero senza segno su x bits corrispondente all’identificativo univoco del processo
* **dim\_nomeProcesso:** intero senza segno su 32 bits indicante la dimensione del nome relativo a quel processo
* **nome\_processo:** stringa di caratteri contenente il nome del processo
* **dim\_icona:** intero senza segno su 32 bits indicante la dimensione in bytes dell’icona
* **icona:** dati relativi all’icona del processo

es:

list\n\r 2 2020 processo1 3 icona1 2036 processo2 2 icona2

Invia al client le info relative al nuovo processo attivato sul server.

* add\n\r processo

Dove:

* **processo:** pid dim\_nomeProcesso nome\_processo dim\_icona icona

Invia al client il pid relativo al processo che è stato appena chiuso sul server.

* remove\n\r pid

Invia al client il pid relativo al processo con il focus

* focus\n\r pid

Il client invia al server i comandi nel formato seguente:

* com\n\r num\_modificatori seq\_modificatori carattere

Dove:

* + **num\_modificatori:** intero senza segno su 32 bits indicante il numero di modificatori relativi al comando
  + **seq\_modificatori:** (opzionale) una o più tra le seguenti stringhe {CTR, ALT, SHI}
  + **carattere:** un carattere tra i seguenti {a / b / c /… / 1 / 2 / 3 / … / backspace / delete / esc / …}

es:

com\n\r 2 CTRALTD

* close\n\r

N.B. I comandi che inviano più dati non prevedono spazi tra i vari dati

# Descrizione Server

Linguaggio di programmazione: C++

Classi:

* Classe principale di gestione server
* Classe secondaria per gestione dei processi: listWatcher

Classe principale:

1. Creazione del socket
2. Creazione della classe listWatcher
3. Attesa di una connessione
4. Recupero della lista processi dalla classe listWatcher e invio tramite comando list al client
5. Gestione modifiche alla lista tramite comunicazione con classe listWatcher

(registrazione agli eventi di gestione della lista nella classe listWatcher)

1. Invio cambiamento tramite comando add/remove/focus

N.B. I punti 5 e 6 controllano che il client non abbia inviato il comando close, altrimenti:

5b. Chiusura della connessione, attesa nuova connessione

# Descrizione Client

Linguaggio di programmazione: C#

Classe principale:

1. Creazione socket
2. Collegamento con il server (gestione indirizzo IPv4, nomeServer e IPv6 (opzionale)
3. Ricezione lista
4. Ricezione focus
5. Visualizzazione statistiche
6. Registrazione listeners per eventi sulla tastiera
7. Aggiornamento info
8. Chiusura

Implementazione:

Il client potrebbe utilizzare un meccanismo con select. La select include nell’insieme di leggibilità sia il socket da cui riceve i comandi dal server, sia lo stdin da cui riceve la combinazione di tasti dall’utente.

* Il keyboardListener registra la combinazione di tasti
* Lo stdin sblocca la select in modo da poter inviare il comando al server

Inoltre il meccanismo della select permette attraverso l’uso di un timeout di aggiornare le statistiche sui processi ogni tot secondi.

L’evento chiusura della finestra scatena invece l’invio del comando close al server.