Server ver. 0.22+

# MALPROJ

## PROGRAMMAZIONE DI SISTEMA (2015-16)

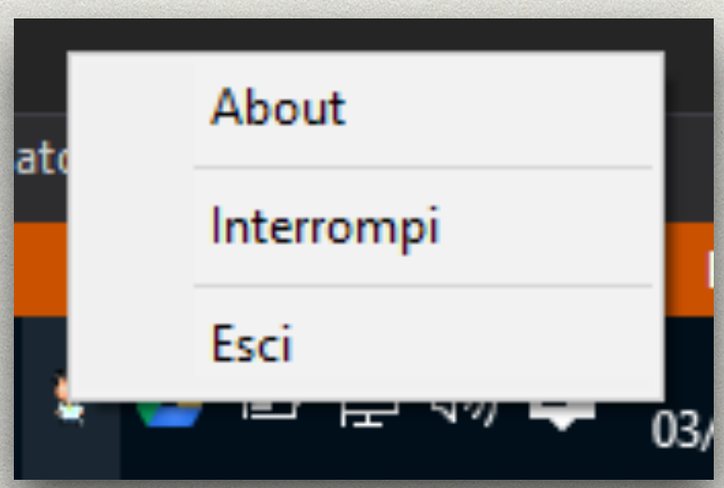
CHIARA LURGO & LUCA MEZZATESTA

# ARCHITETTURA SERVER

# Scelte di progettazione

- Linguaggi: C/C++
- Classe per la finestra grafica: **GWindow**
- 2 thread: **main** & **communication**
- Applicazione localizzata nella **tray area** di Windows

# GUI Features




- **Avvio/Interruzione** del server.
  Se il server è in stop, viene mostrato **Avvia** al posto di **Interrompi**.
  Se è in corso un'operazione, viene mostrato **Attendi...** al posto del nome del bottone premuto.
- **Chiusura** dell'applicazione
- **Connessione automatica** ad un'altra macchina client (se presente) in caso di disconnessione della precedente.
  Un'unica connessione ad una macchina client per volta.

# to Hook or not to Hook?

- **Idea originale**: dll iniettata in ogni processo del sistema operativo che spedisce al server messaggi in base agli eventi della finestra legata al processo.
- **Test**: una hook che stampa su schermo i messaggi che cattura
- **Risultati**: la dll hook può catturare solo messaggi relativi a mouse e tastiera
- **Conclusione**: not to Hook

# Polling

- Ad ogni ciclo:
  - vengono letti i **comandi dal client** (se presenti),
  - vengono registrati i **nuovi dati** delle finestre tramite *EnumWindows(…)*,
  - viene identificata la finestra **in focus** tramite *GetForegroundWindow(…)*,
  - vengono **inviati** i nuovi dati al client

# Processi e icone

- L’ordine delle chiamate a syscall in *EnumWindows*(…) è:

    1. *GetWindowThreadProcessId(…)* — restituisce il ***pID*** data la *hWnd*,
    2. *OpenProcess(…)* — ricava la **handle del processo** dato il *pID*,
    3. *GetModuleFileNameExA(…)* — ricava il **nome completo dell’eseguibile** (formato **ASCII**, da inviare al Client) del processo data la handle del processo,
    4. *GetProcessImageFileName(…)* — ricava il **nome completo dell’eseguibile** (formato **UNICODE**) del processo data la handle del processo (più efficiente rispetto al precedente…),
    5. *SHGetFileInfo(…)* — restituisce le informazioni relative al file. Di queste se ne estrae l’icona **HICON**. Le icone vengono poi salvate in formato *.ico* .

# Comandi

- Il server quando riceve un comando:

    1. Se il comando è **command** avvia la procedura per l'identificazione del comando,

    2. Ricava *hWnd* e *pID*, ricerca la finestra e se la trova la mette in focus (per evitare problemi di temporizzazione)

    3. Inietta il comando tramite *SendInput(…)*

- Si potrebbe anche pensare di salvare la *hWnd* corrente, mettere in focus la finestra richiesta, e poi riportare in focus la vecchia finestra. Feature implementata ma commentata. Vogliamo che l'utente del server veda ciò che sta accadendo.

# PROTOCOLLO DI COMUNICAZIONE

# Lessico

- C = Client
- S = Server
- C -> S = Messaggio mandato da C a S
- *hWnd* = Handle della finestra (per il Client è solo un numero intero!)
- *pID* = process ID

# Comandi (C->S)

- **command** <*hWnd*> <*pID*> <*n_tasti*> <*tasto_1*> …
- (*hWnd*, *pID*) — identificano univocamente la finestra per il server
- I tasti sono V-KEYs di windows
- Se il comando è andato a buon fine, il server risponde con **ok**, altrimenti **error.**
- Il client risponde con **ok**.

# Comandi (C->S)

- **ncommands** <*n_comandi*>'\n'
  <*hWnd*> <*pID*> <*n_tasti*> <*tasto_1*> …'\n'
  …
- Come il precedente, ma con una lista di finestre
- *n_comandi* — numero di elementi nella lista
- Se il comando è andato a buon fine, il server risponde con **ok**, altrimenti **error.**
- Il client risponde con **ok**.

# Nuove finestre (S->C)

- Primo messaggio: **newwindows** <*n_finestre*>
- Altri *n_finestre* messaggi:
  <*percorso_eseguibile*>'\n'
  <nome_finestra>'\n'
  <*hwnd*>'\n'
  <*dimensione_icona*>'\n'
  <*icona* (opzionale)>
- Il client risponde con **ok**.

# Errori icona

- <*dimensione_icona*>
  - > 0 — l’icona esiste
  - = 0 — l’icona non esiste
  - = -1 — l’icona esiste ma c’è stato un errore durante il salvataggio
  - = -2 — l’icona è troppo grande per essere mandata
  - = -3 — non è stato possibile leggere il file dell’icona (0 byte letti da Windows)
  - = -4 — errore non specificato

# Finestre chiuse (S->C)

- **closed 1**'\n'
  <*hWnd*> <*pID*>'\n'
- In origine le finestre chiuse venivano mandate in un unico messaggio
- Per facilitare le operazioni svolte dal client, vengono mandate una per volta
- Il client risponde con **ok**.

# Focus

- **focus** *<hWnd> <pID>*
- **focus nullwindow** — nessuna finestra sta tenendo il focus (momento di transizione)
- **focus windowsoperatingsystem** — il focus è tenuto dal sistema operativo
- Il client risponde con **ok**.

# Chiusura connessione (C->S)

- **stop**
- Il server risponde con: **stop**
- Nel caso di un errore durante la comunicazione, il client o il server interrompe automaticamente la connessione.