Mohammad Adeel Hassan
07277

SP Final Project

Architecture

The program consists of 3 parts

- the Task Manager – manages the processes of a client
- the Client
- The Server

Each of these is in a separate executable.

Task Manager (TM)

Maintains a list of processes (implemented as an array).

Communicates through the following channels:

- A socket to the client
- A pipe for receiving commands from the server
- A pipe for receiving results from the server
- A pipe for sending commands to the server
- A pipe for sending results to the server

Uses multiplexed i/o to listen on the input pipes.

Client

Communicates with the TM using a socket.

Uses multiplexed i/o to simultaneously listen on the socket and stdin.

Server

Maintains a list of clients.

Uses multiplexed i/o to simultaneously listen for incoming connections, stdin input and the cmd and result pipes for each client.

Mohammad Adeel Hassan
07277

Commands

Client

conn[ect] <hostname> <port>

Connect to the server.

disconnect

Close the TM and the socket. Keep the client running.

Client -> TM

sleep <seconds>

Sleep for <seconds> seconds. (Can be used to check the non-blocking behavior of the client).

[run] <program-name> [<count>]

Start <count> instances of the program.

list

List alive processes.

list all

List all alive or dead processes started through the TM.

list details

List all alive or dead processes started through the TM, along with their start and end times and the elapsed time.

kill [<pid> | <process-name>]

Kill process by pid or name

kill [all | *]

Kill all processes

add [<num1> [<num2> …]]

sub [<num1> [<num2> …]]

mul [<num1> [<num2> …]]

div [<num1> [<num2> …]]

Add, subtract, multiply or divide.

exit | ex | quit | q | disconnect

Exit the TM.

Mohammad Adeel Hassan
07277

Client -> TM -> Server

msg <message>

Message shows up on the server terminal.

Server

list

List currently connected clients.

disconnect <ip>:<port>

Disconnect this particular client.

disconnect all

Disconnect all clients.

exit | ex | quit | q

Disconnect any connected clients and exit.

Server -> TM -> Client

broadcast <message>

Message shows up each client's terminal.

cl <ip>:<port> <command>

Send command to this client's TM and receives the output.

Mohammad Adeel Hassan
07277

Working

Server

The server starts, binds to a port and waits for connections, while at the same time listening on stdin.

When a client connects, it

- fork-and-exec's a new TM, makes 2 input and 2 output pipes to it,
- creates a new client structure, populates it,
- adds it to a linked list of such structures,
- starts listening on the 2 input pipes (cmd and result) to that TM.

Client

The client starts and waits for user input.

On receiving the connect command, connects to the server on the ip and port provided.

From there onwards, forwards user commands to the associated TM, and displays the TM output on stdout.

TM

Listens on the client socket and the 2 pipes to the server, simultaneously.

Depending on the command, decides the appropriate input fd and output fd. Then reads from the input fd and writes to the output fd.

Limitations

- TM does not support command line arguments for the programs it creates
- No client-to-client communication

Unique Features

- Can execute any command on a client's TM from the server
- Can send a broadcast message to all clients from the server
- Client and server detect when the other end of the socket closes
- I/O multiplexing avoids the overhead of multi-threading
- Client-to-client communication should be fairly easy to implement