SP Project Report

Client/Server model

Usman Ahmed Saeed

17857

**Manual:**

Client-side code will run by giving two additional command line arguments, the first being IP address of the server, and the second being the port number which is fixed at 50000. Client has the option of using 11 commands (add,sub,mult,div,kill pid, kill name, list, list all, run name,exit);

Server-side code will run by just executing the executable file such as ./testing. The server has the option of using 4 commands (list, list serialno, print message, print serialno message).

Both of the files are made in C++, and on visual studio so additional work of changing the permission of the executable is needed to be done to run the executable file.

**Architecture:**

Client:

There are two threads functioning, one being the main thread, and other being the thread that was created which receives the output from the server. The main thread reads from the user, and sends the command to the server through the designated socket.

Server:

Here we have created 3 list, one being the active process list of each client, and one being the process list (terminated process included) of each client. The third list is the global list which maintains client information of each client that is being connected to the server.

To handle multi-client connections, the server calls fork where each client is handled by the new child process. This process handles the incoming connections from the client and process it according where each command is divided in to its method respectively.The parent process returns to accept to wait for new connections. In addition there are two threads that are being created on the server-side code, the first thread being the serverinput thread which is responsible for the inputs given at the server terminal such as list or print. Another thread is the clienthandler which is made for each client. The serverinput thread has the information of all the clients connected with the server, whereas each clienthandler thread has the information about the its respective active process that are active. Both of them are linked through a pipe where serverinput thread triggers the clienthandler(which is on blocked due to read from the pipe) using the pipe, which then prints the active process list of the client. Another command that is also handled by the serverinput thread is to send messages to the client whether to all client or any specific client.

**Limitation:**

Due to the server-side being serialized, if any program is terminated from outside, or crashes, the list is not updated until the client sends any command to the server, as the server is stuck on the read for the client’s new command. This condition cannot be handled in the signal function as the process list is not global in my case.

**Bonus details:**

When a specific process is terminated from the outside, the server sends the respective name of the process to the invoking client informing them that this particular process have been terminated from the server side. Due to the server-side being serialized, until any command is not sent from the client to the server, the client cannot be informed about the terminated process.