SP Project - Client Server Model

Report

Hasan Imran | 11459

HELP:

Commands for client:

-> conn [IP] [port]
-> disconn

-> add num1 num2 num3 ...
-> sub num1 num2 num3 ...
-> mul num1 num2 num3 ...
-> div num1 num2 num3 ...

-> help

-> run [process name] [file to open by the process]
-> list | [all]

-> kill [pid | process name | all]

-> exit

Commands for server:

-> list [clients | processes]

ARCHITECTURE:

Client:

-> client has one thread for reading from screen and sending to the server and a main thread which reads from server and writes to the screen.

Server:

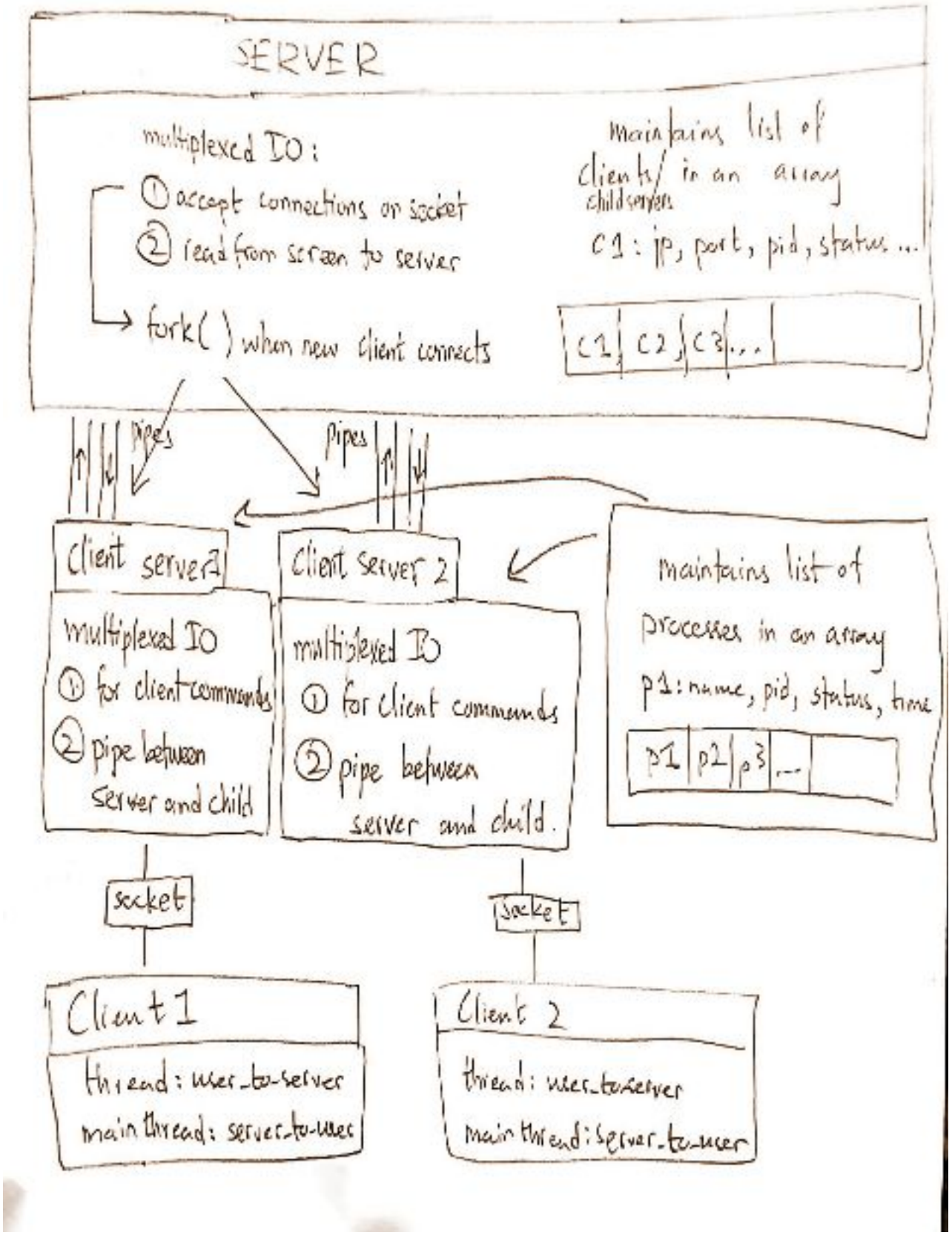
-> i have used a multiplexed IO approach.

-> the main server waits for input from the socket to establish a connection or for the user to input a command

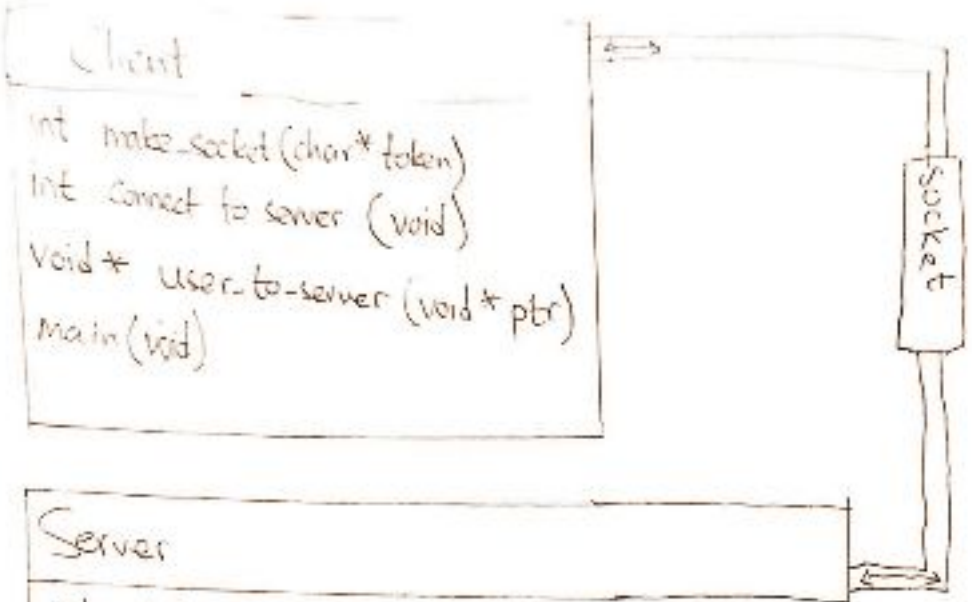
-> a child server is created to deal with a client after a connect has been established.

-> the main server interacts with the user and its child server using pipes.

ARCHITECTURE:



Basic Class Diagram



Server

Struct list → for processes
Struct list2 → for clients.

Static struct list procen_list
Static struct list2 client_list
static int pointer
Static int no. of clients.

Static void sig_handler → for processes
Static void sig_handler → for clients
static void math (char* token, int fd)
Static void run (char * token, int fd)
" " display_list(" , ")
" " kill_process (")
Static int make_socket_server (void)
Static void display_list_client (char * token, int fd)
int main (void)

LIMITATIONS:

-> process gets killed but list doesn't get updated.

reason: signal handler does not queue the same signal more than once. so in kill all many sigchld get generated but only one goes through.

**handled it by changing the status to 0 after calling kill all only

-> when we exit server abruptly(SIGINT), if client has not been disconn or closed then it continuously prints the enter instruction in the client terminal

-> when we exit client abruptly(SIGINT), if server has not been disconn or closed then it continuously prints the prev instruction in the server terminal.

-> have not implemented write return value check for perror

-> have used the exit() command after perror where there is no while loop as to stop any unpredictable behaviour of the program.