CLIENT AND SERVER1

Develop a TCP sequential server (listening to the port specified as the first parameter of the command line, as a decimal integer) that, after having established a TCP connection with a client, accepts file transfer requests from the client and sends the requested files back to the client, following the protocol specified below. The files available for being sent by the server are the ones accessible in the server file system from the working directory of the server. Develop a client that can connect to a TCP server (to the address and port number specified as first and second command-line parameters, respectively). After having established the connection, the client requests the transfer of the files whose names are specified on the command line as third and subsequent parameters, and stores them locally in its working directory. After having transferred and saved locally a file, the client must print a message to the standard output about the performed file transfer, including the file name, followed by the file size (in bytes, as a decimal number) and timestamp of last modification (as a decimal number).

The protocol for file transfer works as follows: to request a file the client sends to the server the three ASCII characters "GET" followed by the ASCII space character and the ASCII characters of the file name, terminated by the ASCII carriage return (CR) and line feed (LF):

G	E	T		...filename...	CR	LF

(Note: the command includes a total of 6 characters plus the characters of the file name). The server replies by sending:

+	O	K	CR	LF	B1	B2	B3	B4	T1	T2	T3	T4	File content.....

Note that this message is composed of 5 characters followed by the number of bytes of the requested file (a 32-bit unsigned integer in network byte order - bytes B1 B2 B3 B4 in the figure), then by the timestamp of the last file modification (Unix time, i.e. number of seconds since the start of epoch, represented as a 32-bit unsigned integer in network byte order - bytes T1 T2 T3 T4 in the figure) and then by the bytes of the requested file.

To obtain the timestamp of the last file modification of the file, refer to the syscalls *stat or* f*stat.*

The client can request more files using the same TCP connection, by sending many GET commands, one after the other. When it intends to terminate the communication it sends:

Q	U	I	T	CR	LF

(6 characters) and then it closes the communication channel.
In case of error (e.g. illegal command, non-existing file) the server always replies with:

-	E	R	R	CR	LF

(6 characters) and then it closes the connection with the client.

SERVER2

Develop a concurrent TCP server (listening to the port specified as the first parameter of the command line, as a decimal integer) that, after having established a TCP connection with a client, accepts file transfer requests from the client and sends the requested files back to the client, following the same protocol used in Lab exercise 2.3. The server must create processes on demand (a new process for each new TCP connection).
When developing this new server, use the same directory structure already used for the client and server of Lab exercise 2.3 (folder `lab2.3`). Copy the files named `test2.sh` and `makezip.sh` (provided in the `lab3.1` folder of the zip archive where you have found the material for Lab 3) to folder `lab2.3` and, inside `lab2.3/source`, create a new directory named `server2` (it will be a sibling of `server1` and `client1`). Write the code of your new server inside this directory (you can use the libraries you already have placed in the `source` folder).