In this project we did only the base extension.

server.c:
In server.c we set a a main socket that persisently listens to all incoming connections after a connection is sensed
it created a worker thread to decode and execute that action. Incoming commands are given in the form of

```
r(4(6
command/filedescriptor/bytes to read

c(8
command/filedescriptor

w(10(aesdf(5
command/filedescriptor/bytes to write/length of string

o(test.txt
command/file path
```

Every bit of important information is seperated by the delimiter '(' . The command type is always the first in
message that is sent over followed by the file descriptor in most cases.

Given the string we then extract the parameteres accordingly and carry out functions on the local machines that access
the files. Depending on what is returned from those functions we pass back information about the success/failure of each functoin or the bytes read/writen, or in the case of read a string.

After message is send back we free the sock and return thus exiting the thread that we created. All while this is going on the server
is listening for more commands from other clients in the main() function.

libnetfiles.c:
In libnetfiles we had the defeninitons from every function such as netopen,netclose,netwrite,netread,netserverinit. These fucntions
all start the same by creating a socket that will connect to the main socket from the server. In each funcation we take the nessceeary parameters and make the string as seen above that will be sent to the server to decode. When we get back information from the worker
thread we then decode that message to decide what the function will return to and whether or not to indicate a success or failure.

libnetfiles.h:
In the header file we used text replacement to define all net functions and linked them to libnetfiles.c.