PARCIAL-protocolos de internet

Realize un programa cliente servidor TCP ITERATIVO

**Comandos para servidor:**

Gcc servidor.c -o s

./s

**Comandos para cliente:**

Gcc cliente.c -o c

./c ParcialTL2020 msj cueva-151620542

Msj-hola

Msj-comova

Msj-log

Msj-fin

***. Para iniciar otro cliente:***

./c ParcialTL2020 msj cueva-151620542

Socketcalls utilizadas:

Servidor:

.socket(),bind(),listen(),accept(),recv(),send()

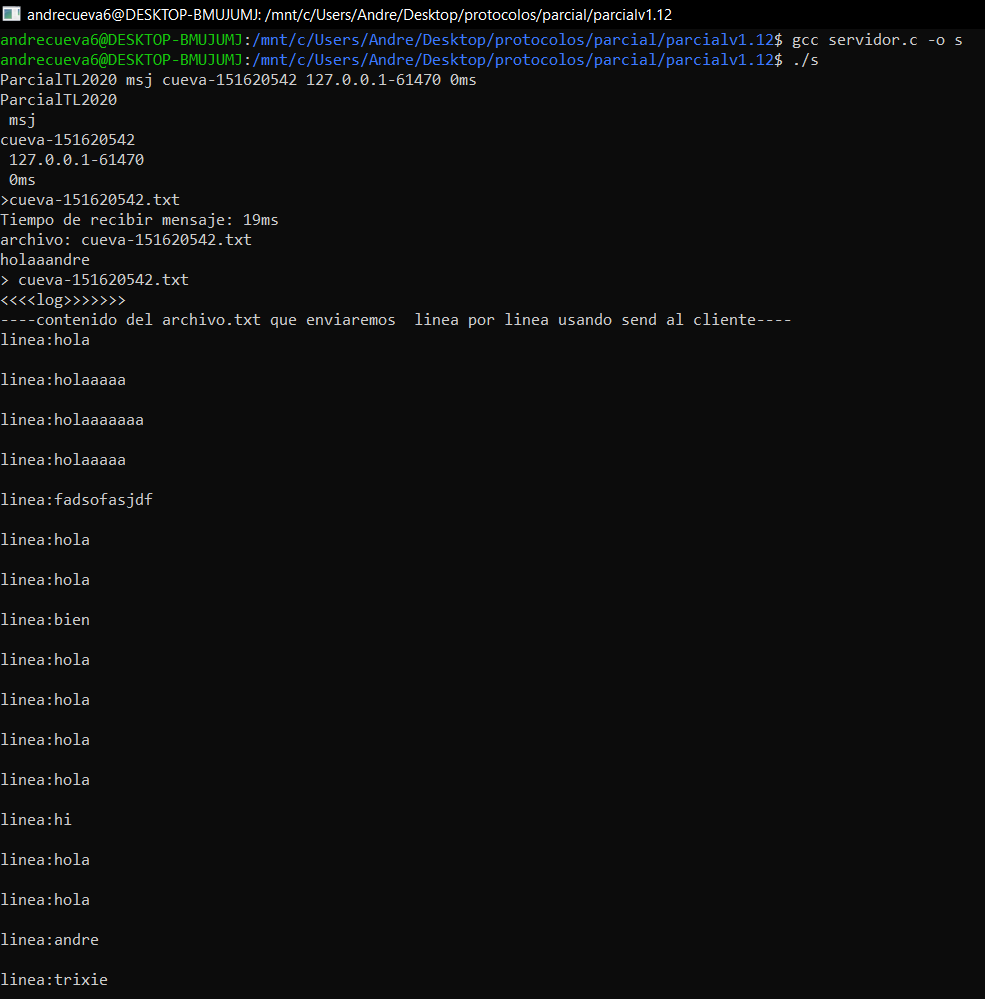
Cliente:

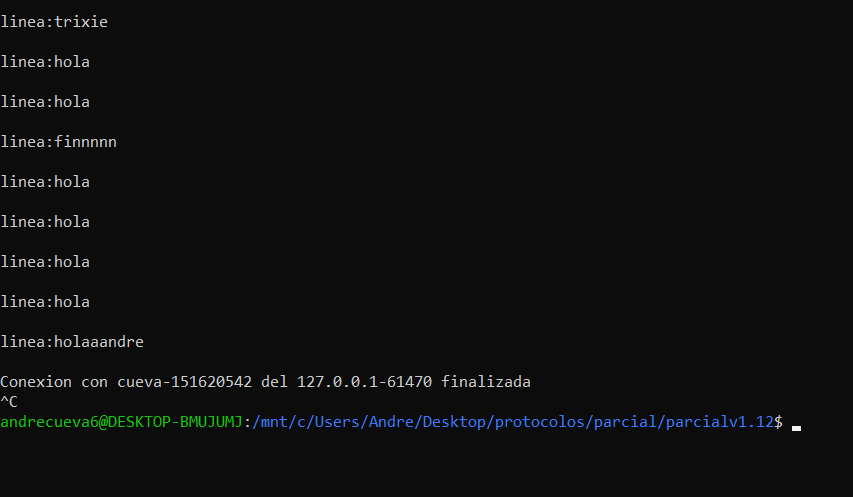
.socket(),connect(),recv(),send()

El funcionamiento de cada socketcall esta escrito en cada archivo c correspondiente a su uso.

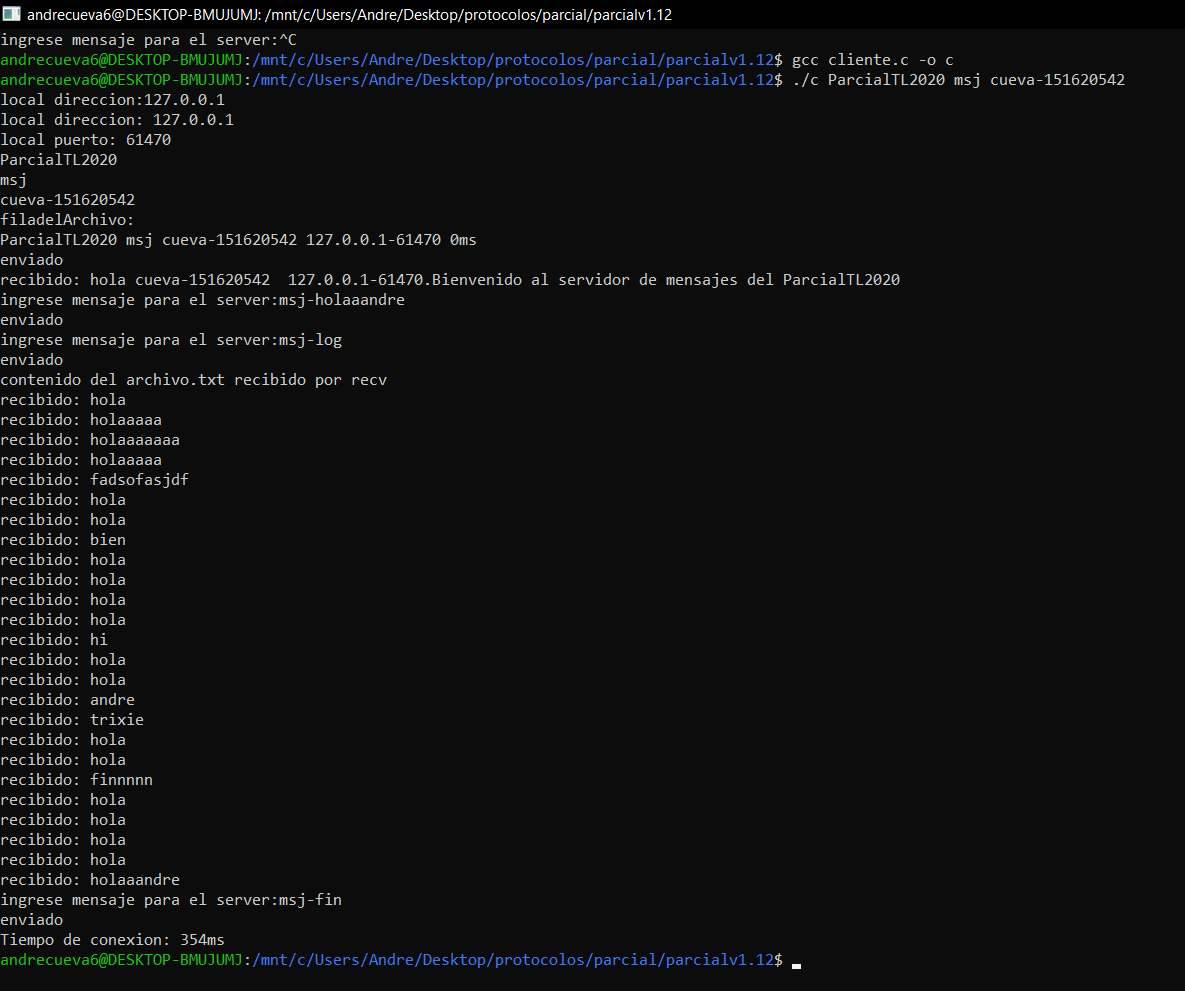
**A continuación colocare imágenes de como funciona:**

**Servidor(hoja siguiente):**





**Cliente:**



**Funcionamiento que realice:**

Considere que el cliente manda solicitud al inicio por comando cuando abre el archivo ./client … Luego en el código se verifica, coloque el tiempo de conexión total del cliente que al final cuando use el comando msj-fin dira el tiempo total de conexión del cliente. Los comandos validos son msj-“loquequieramandaralchivodeesecliente” con este comando guarda el eso en el archivo.txt correspondiente a ese cliente y se desconecta y abre denuevo eso sigue guardado debido que el archivo de ese cliente ya esta creado y no se elimina. El otro comando que se puede usar es el msj-log que mostrar todo el contenido del archivo.txt correspondiente a ese cliente usando un ciclo while que terminara cuando sea EOF el archivo del cliente, esto lo hice con un recv y send en bucle. El tiempo e igual de importante es el msj-fin que lo que hace es terminar con ese cliente y mostrar un mensaje en el servidor que indica que la conexión termino. El msj-fin muestra que se hizo un shutdown para terminar esa conexion de ese cliente desde el lado del servidor. Msj-fin cierra la conexion de es cliente unicamente puede haber otros clientes.

**El funcionamiento esta escrito de tal manera que si lees el código c de cada programa se va a poder entender ya que ambos programas estan comentados en su totalidad. Cuando hay un socketcall que utilice de nuevo no realice comentario porque era algo redundante.**