Stream & Socket

Intro to stream

-When the data is too large to be processed all at once, we can distribute the data into each chunk. Mainly used in movies or multiplayer games. The same applies if the data is infinitely expanded.

-The buffer has a maximum limited capacity of 65536. When the data source fills the first buffer, it will continue to fill the next buffer, and the filled buffer will become a chunk and enter the observers.

1. Demo- buffers and streams
    1. The first demo uses the fs functionality to read and write through the stream using createReadStream (readable stream) and createWriteStream (writeable stream). The first input file has 143327 bytes.
    2. source.on("data",callback function) callback function will return how many chunks created.
    3. So the stream will divide the total byte into three parts because each chunk can have 65536 bytes. A function is required to end after they are read or written.
2. Demo-stdin
    1. In this demo, it shows that after creating an output document, we can pass input data, and the data will be written to the document.
    2. process.stdin.on allows the user to input data and convert it into chunks.
3. Demo-file copy
    1. The first file create-1gb is just to create a buff of size 1gb in the input file. So the steps are similar to the first demo. Since there is already a 1gb file we need to access, we need to read the data into a chunk and then write the output file. In addition, the ability to see the percentage progress has been added.
4. Demo-combine
    1. In this demo, it uses synchronously similar to assignemnt1. It is done by using recursive function. The rest of the process is the same as the previous demo.
5. Demo-pipe
    1. So pipe is very useful when just putting data to data. He can't transfer chunks.

How to use socket

-A socket connects a server and a client by creating pairs of codes.

-net.createServer is used to create the server, and net.createConnection({host,port}) is used to connect to the server.

-On the server code, the server will listen to the connection and trigger connection_handler(callback).

-The server also has a listen function, which port to listen to.

-On the client, the user will create a socket to create a connection (createConnection), and its parameters are port and host.

-socket will also listen connection and callback function (connect_handler).

-In order to run this code, it needs to be run in multiple PowerShell.

-The client should not have the ability to close the server under any circumstances, so we should use the four-way handshake to close the connection whenever possible. We also need to use socket.destory to delete erroneous data when disconnecting.

1. Demo-server and client
    1. Client and server will establish tcp connection through three way handshake.
    2. Briefly describes how to get the client to connect with the server.
2. Demo-resistant-servers
    1. To implement the four way handshake, both client and server will listen error or end. Listen error is to prevent the connection from being interrupted unexpectedly. Listen end is so that when the server sends the end, the client can also respond and complete the four way handshake.
3. Demo 3
    1. The data handler is mainly added, so every time the server gets readable data, it will trigger the data handler, and the data will be calculated in a chunk. Every time the client sends a hello to the server, and the server returns the world to the client.