Starry Node.js Code Challenge

Your challenge is to connect to a TCP server at `code-challenge.starry.com:1337`, receive a list of files, and download each of them over a *single persistent socket*.

Communication with the server is accomplished via length-prefixed messages. The format of each is described in the `Message Formats` section below.

The completed challenge should only use the core Node.js APIs, no third-party dependencies.

Please do not share this exercise or post your solution to a public repository.

Project outline

You will connect to a TCP server located at `code-challenge.starry.com`, on port `1337`.

After establishing a connection, you will send a `LIST_DIRECTORY` message (as defined below) requesting the contents for a directory named `content`.

Next, you'll receive a `FILE_LIST` message whose payload is a string containing a list of comma-separated file names (e.g. `'content/foo.txt,content/bar.txt'`).

For each file, you will send a `GET_FILE` message to request the file contents.

For each `GET_FILE` message send, you will receive a `FILE` message in response, which will contain the raw binary file contents.

Upon receiving the contents for each file, your goal is to write each file to a directory on your machine and email your solution back to Starry.

Note: you do not need to send the downloaded files, just the source code.

Streaming

TCP is a byte stream protocol, which means that "messages" can arrive at the client in multiple chunks. In Node.js terms, that means that the client socket may receive multiple `data` events for a single message. Your application must anticipate this, and stitch the chunks back together in order to form a complete message.

This can be accomplished by reading the `length` field at the beginning of each message.

After reading `length`, your parser should expect `length` additional bytes prior to considering the message to be complete.

Note that `length` will refer to the byte-length of the `payload` field plus an additional byte for the `type` field.

Message Formats

LIST_DIRECTORY

Request a list of files in a given directory.

Sent by a client, and will receive a `FILE_LIST` message in response.

Name	Byte Offset	Format	Description
Prefix	0	utf-8 string	Always the string "MESSAGE"
Length	7	32-bit unsigned integer (big-endian)	The length, in bytes, of the rest of the message
Type	11	8-bit unsigned integer	The message type (always 0 for LIST_DIRECTORY messages)
Payload	12	utf-8 string	The directory name

FILE_LIST

A comma-separated list of files in a directory.

Sent by the server in response to a `LIST_DIRECTORY` message.

Name	Byte Offset	Format	Description
Prefix	0	utf-8 string	Always the string "MESSAGE"
Length	7	32-bit unsigned integer (big-endian)	The length, in bytes, of the rest of the message
Type	11	8-bit unsigned integer	The message type (always 1 for FILE_LIST messages)
Payload	12	utf-8 string	Comma-separated string of file names within a directory

GET_FILE

Request the contents of a given file.

Sent by a client, and will receive a `FILE` message in response.

Name	Byte Offset	Format	Description
Prefix	0	utf-8 string	Always the string "MESSAGE"
Length	7	32-bit unsigned integer (big-endian)	The length, in bytes, of the rest of the message

Type	11	8-bit unsigned integer	The message type (always 2 for GET_FILE messages)
Payload	12	utf-8 string	File name

FILE

The contents of a given file.

Sent by the server in response to a `GET_FILE` message.

Name	Byte Offset	Format	Description
Prefix	0	utf-8 string	Always the string "MESSAGE"
Length	7	32-bit unsigned integer (big-endian)	The length, in bytes, of the rest of the message
Type	11	8-bit unsigned integer	The message type (always 3 for FILE messages)
Payload	12	buffer (i.e. binary data)	Raw file contents

ERROR

A generic error message.

Sent when the server receives an invalid message.

Name	Byte Offset	Format	Description

Prefix	0	utf-8 string	Always the string "MESSAGE"
Length	7	32-bit unsigned integer (big-endian)	The length, in bytes, of the rest of the message
Type	11	8-bit unsigned integer	The message type (always 4 for ERROR messages)
Payload	12	utf-8 string	Error message