Go-Client&Server

Last updated by | Hüseyin Polat Yürük | 26 May 2022 at 09:07 CEST

Binalyze Coding Challenge for Go

Client&Server

You are assigned to create the following project in Go programming language with version 1.16.

The project is about creating two applications to demonstrate you have enough knowledge about networking, data types, and concurrency.

The first application is a CLI application (Client) that sends file information to a server application (Server) over TCP/IP with a custom message described below. The client accepts the server address and its ID as command line arguments when started.

The client walks all the regular files in the file system starting from the root directory for Linux and all Volumes for Windows. It takes the file's absolute path, file's size, and file's permission bits for every file and packs the data into a message to send it to the Server along with message size, client id, and checksum.

Assume that the Server does not ingest incoming data as fast as Clients produce. Because of this, you should have an in-memory queue implementation to let the maximum N number of messages can wait in the queue, and the Client must pause walking and wait for queue availability. Any error encountered during walking should be logged appropriately.

The server accepts the Client's messages over TCP/IP and listens to port 7654 always. Every read valid message from clients is written to a known file. Since messages are sent in binary format, they must be decoded and written to the file in CSV format. Communication is one way and the Server does not send any message to the clients.

Message Format

```
// | Message                                                                                             |
// |-----------------------------------------------------------------------------------------------------|
// | Message Header           | Message Body                                                | Checksum   |
// |-----------------------------------------------------------------------------------------------------|
// | Message Size | Protocol  | Client ID | File Size | File Permissions | File Path Length | File Path |              |
// |-----------------------------------------------------------------------------------------------------|
// | uint64       | 1 byte    | uint32    | uint64    | uint32           | uint32           | chars     | uint32     |
// | 8 bytes      | 1 byte    | 4 bytes   | 8 bytes   | 4 bytes          | 4 bytes          | variable  | 4 bytes    |
// |-----------------------------------------------------------------------------------------------------|
```

- Byte Order must be Big Endian for every field.

- There is only 1 protocol for now, and its value is 0x00.
- Message checksum must be calculated using CRC-32 IEEE Polynomial for Message Header and Message Body together.
- Server should write the incoming messages to the file with the first in first out rule.
- There must be only one project for the client and server. Prefer cmd/client and cmd/server folders for main applications.
- Creating a Makefile (or a script) to build and test is a plus.
- Creating a README.md is a plus.