CS23820 Assignment

Inglenook Sidings

Dimitar Plamenov Velikov

Email:dpv@aber.ac.uk

Document date: 16.12.2016

Introduction

This report will discuss the implementation of the classical problem – Inglenook sidings.

The aim of the task is to produce a train consisting of a specific sequence of wagons containing particular goods. The wagons are stored in sidings.

System design

The solution of that task is split in two programs:

- Signal Box Program (**SBP**) - written in C
- Siding Logistics Program (**SLB**) - written in C++

SLB has seven class files and six header files. It holds all the information about the sidings and the wagons. The **main** class **instantiates railway_Iinitializer** and **SBP_communication** objects. **railwayInitializer** reads a text-file - containing data about 10 wagons, then populates a vector with each of them. Afterwards it reads another file which contains data about sidings and creates a vector of sidings. It also assigns the wagons with matching serial numbers to that vector of sidings. Another way of implementing the sidings could be using an array. The number of sidings is fixed such as the size of the array, but there is not a case in this software where the number of sidings in the SLB is extended, so vector has been considered for appropriate data type for storing sidings. **SBPcommunication** takes the vector of sidings and uses **boost asio** to connect and communicate with a server. A vehicle class has been provided along with the assignment diary to provide a basis for building a subclass(es). Polymorphism has been fully implemented using that class as a base class. No changes have been made to that class. **Dynamic cast** has been used in order to stream the vehicle object.

SBP contains four source and three header files. The purpose of **main.c** is to call a function from **server.c** which starts the server. The server receives commands from **SLB** which are passed to functions located in **movement.c**. The commands are processed in **movement.c** and returned back to the server which is sending them to the **SLB**. The header files are holding **function** and **macro definitions** as well as **library imports**. **SBP** only contains information about the size of each siding and the number of wagons it has. This information is stored in **an array of structs**. Each **struct** has **size** and **freeSpace.** Another way of implementing this could be storing number of wagons instead of **freeSpace**.

System operation

- The server has to be started. When it is started It waits for a connection.
- The client has to be started. A file containing wagon positions has to be selected.
- The server and client has to establish a connection.
- The client has to send a **config** message, specifying the number and the size of each siding
- The client has to send a **load** message, specifying the number of wagons in each siding
- At this point the client is able to request movement of wagons between the sidings.

- At any time the client can exit itself using **quit** or exit both programs using **]!**
- At any time, the client can request changing of the sidings stored in the server using **config**

Style guide adoption

I have endeavoured to conform as much as possible to style guides when coding both programs. SBP conforms to the Indian Hill style guide and SLB conforms to the C++ Google Style guide. Part of the adoption includes the naming of files, classes, functions and variables naming. It also includes appropriate usage of newlines, whitespace, statement's brackets and, in the header files, the order and naming of imports and macros.

Additional features

The additional features part of this assignment has been completely implemented. While implementing, it was separated in different parts:

- SLP has to be able to read files with different number of sidings and wagons.
- SBP has to be able to change the size of its array of structs – dynamically allocated memory has been used for this part. SBP **allocates /reallocates** a memory when receives a command which contain **config.** After receiving **exit_value** message the server **releases** the allocated memory and exits.
- SBP – When a connection with a client is closed – it has to look for a new connection. This means that the sever can be used by multiple clients (not at the same time). To accomplish this a code to request socket address reuse was applied.

Conclusion

I believe that I've provided a complete solution of this assignment. I've learned the basics of writing a software in both C and C++ programming languages and I've learned much about socket programming using POSIX and BOOST libraries.