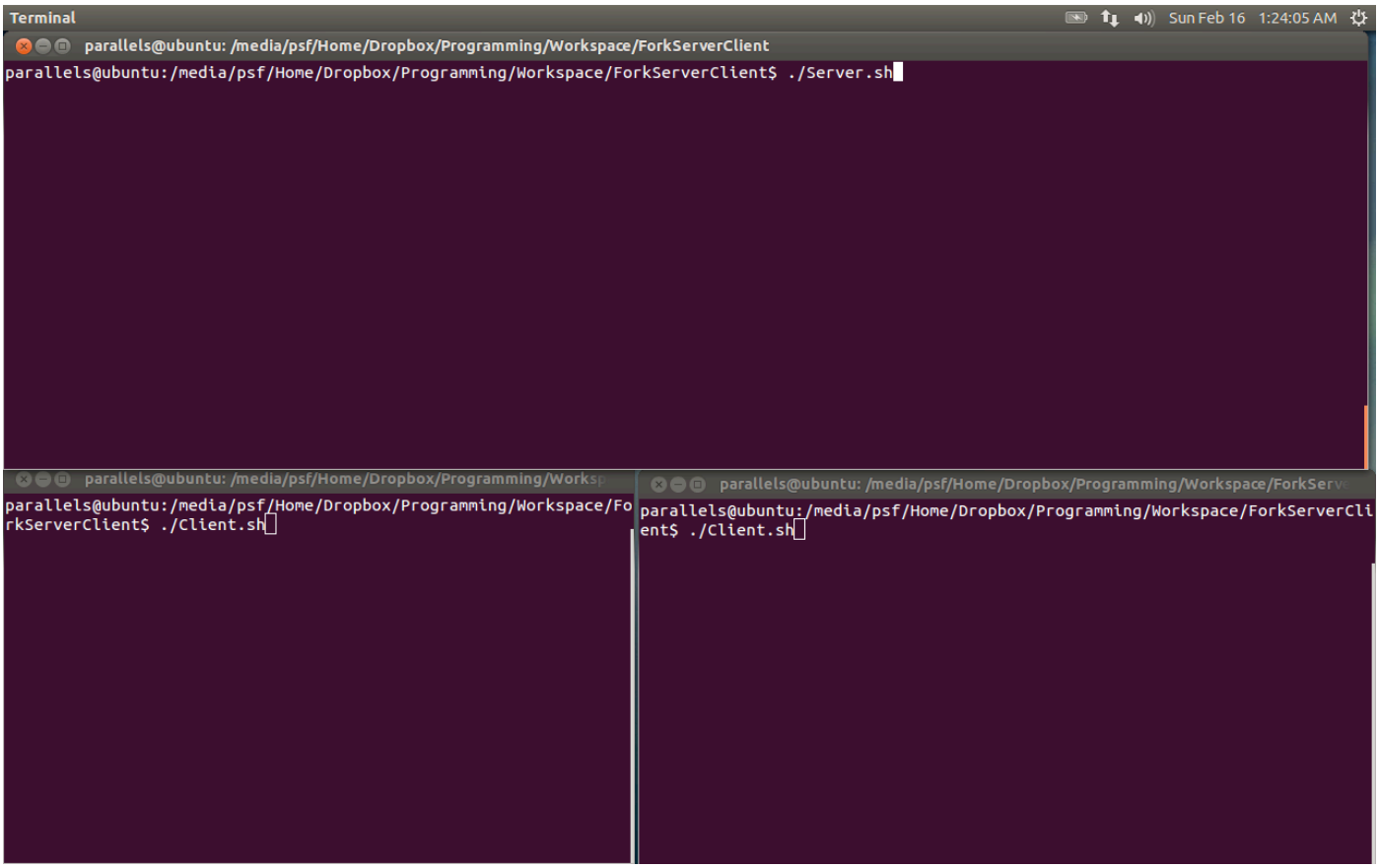
(Interpreted) requirements of the program

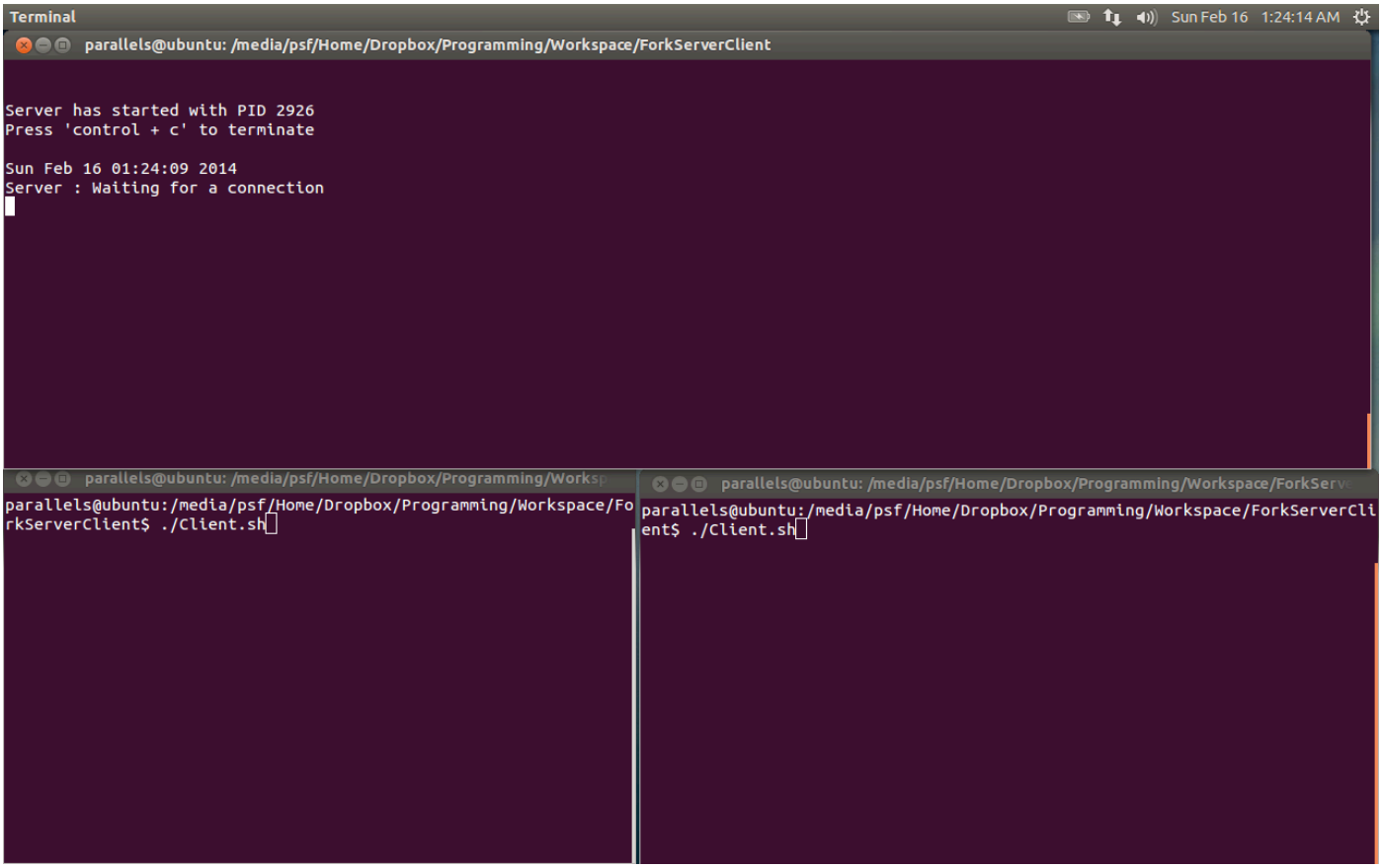
- Create a server program that reads data
- Server forks out a child process upon accepting connection request from a client program
- Create client program that prompts user input
- If user input not found in server, error shown
- Else data found in server is shown

Diagram / Illustrations of program design

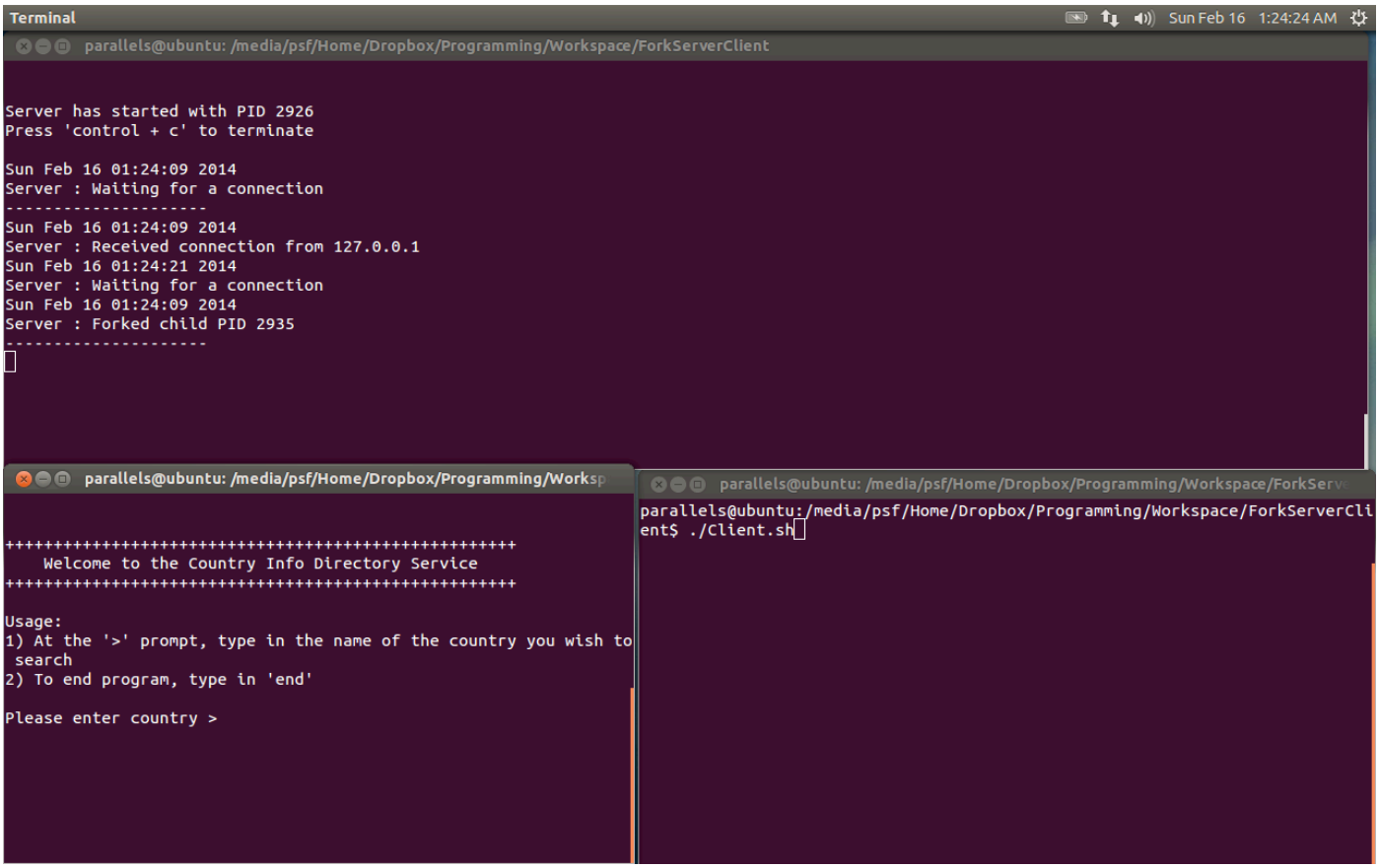
Main features



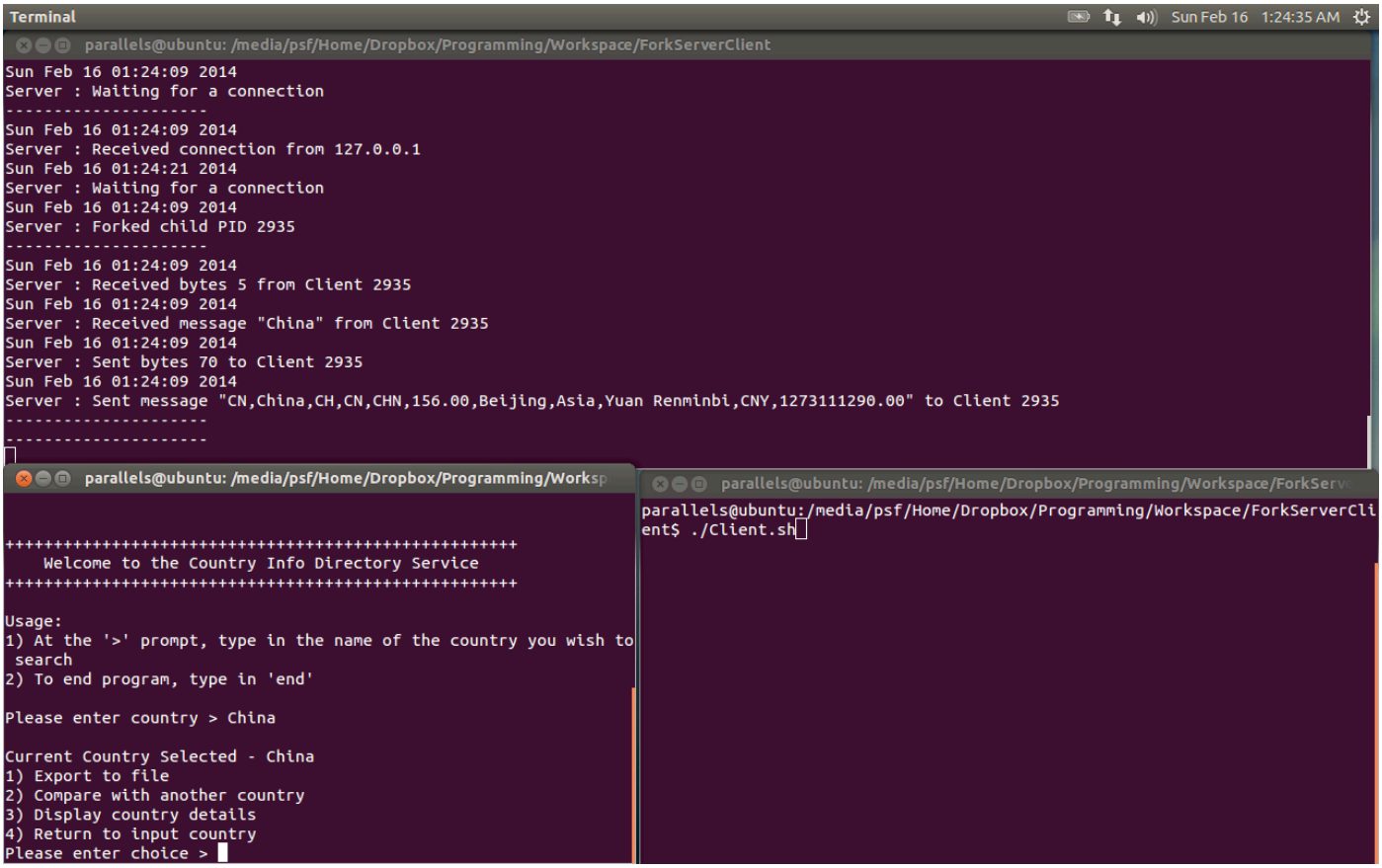
- Before starting up of programs, which are initiated by shell script files



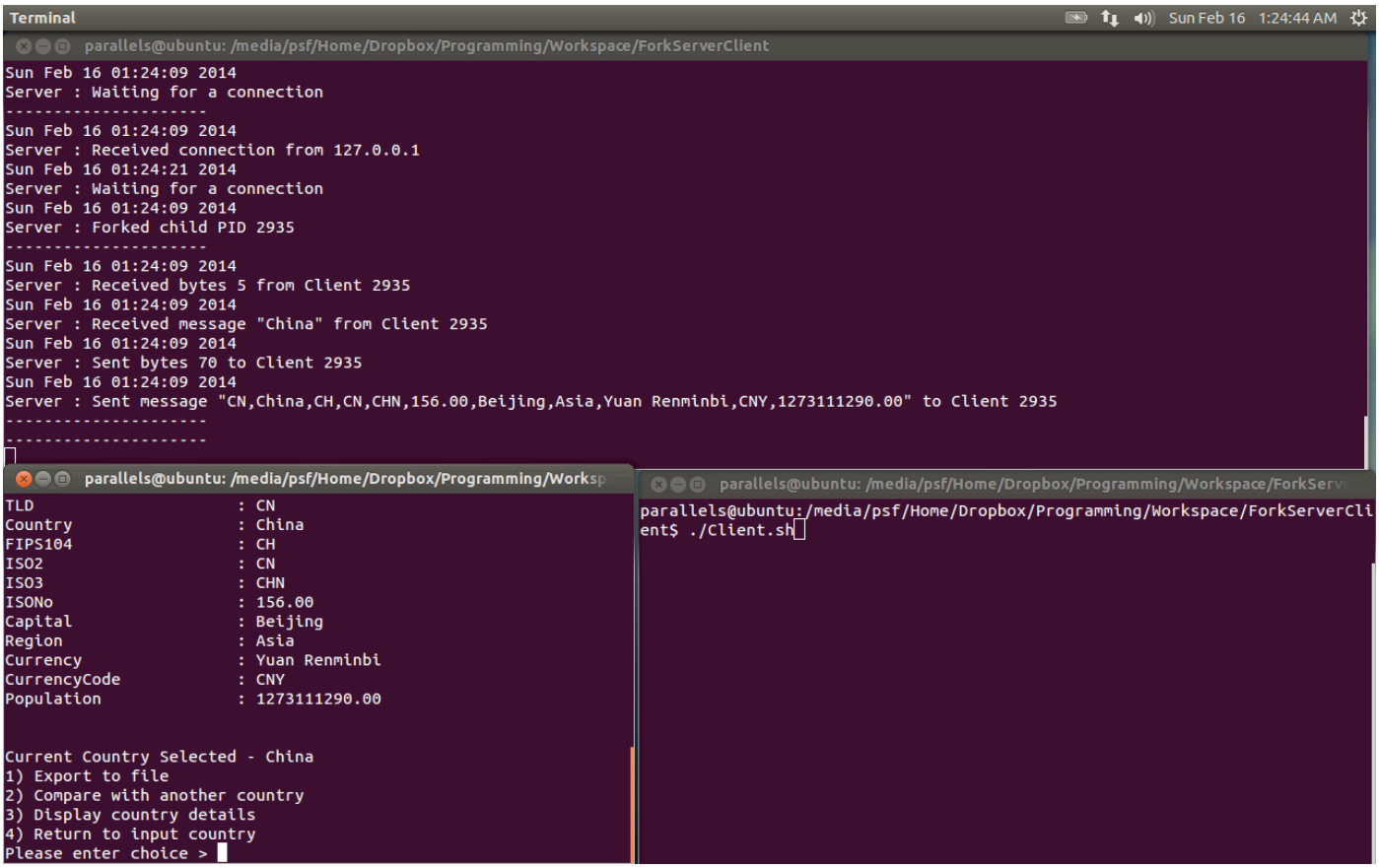
- Server is started



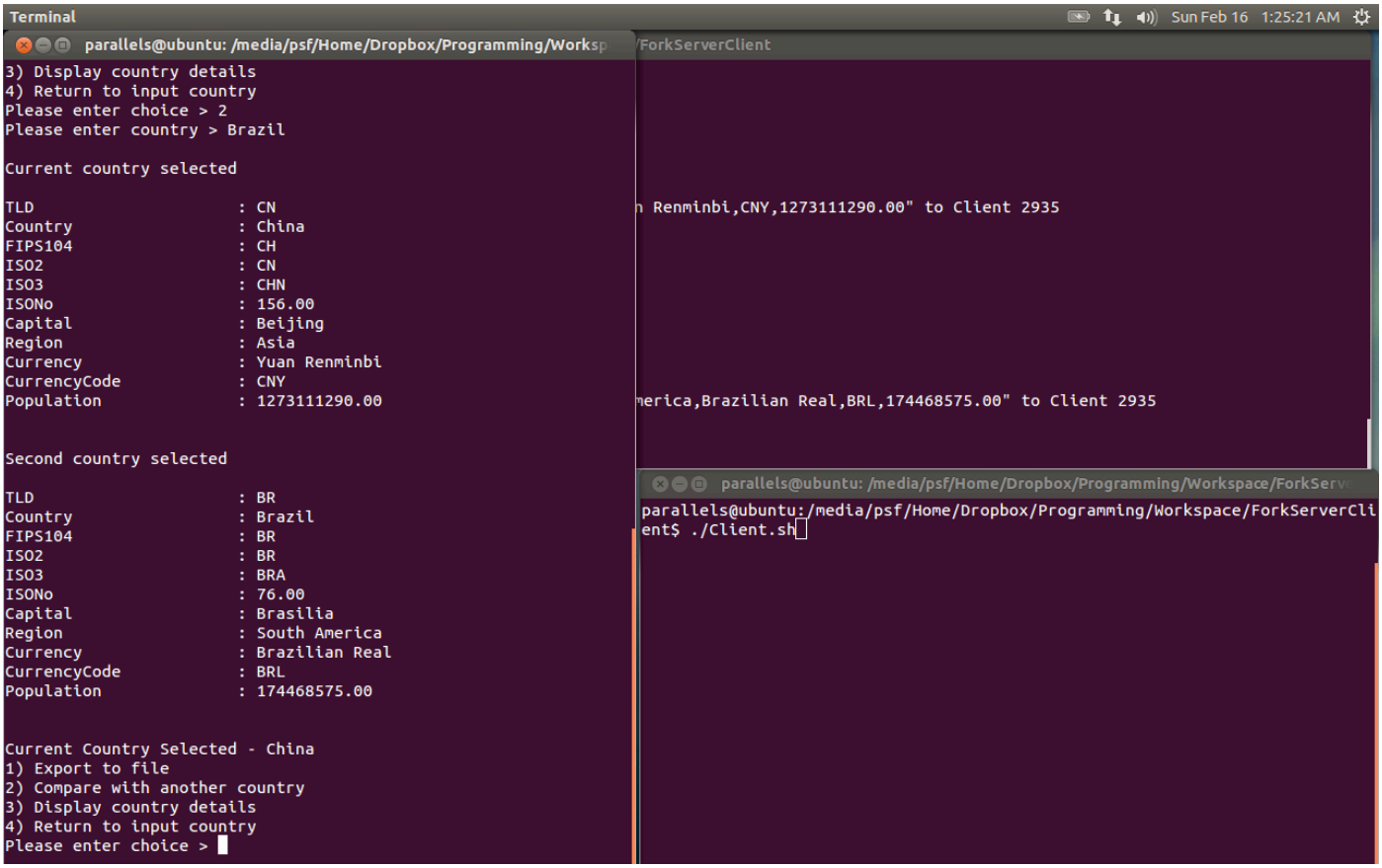
- A client program has connected to server



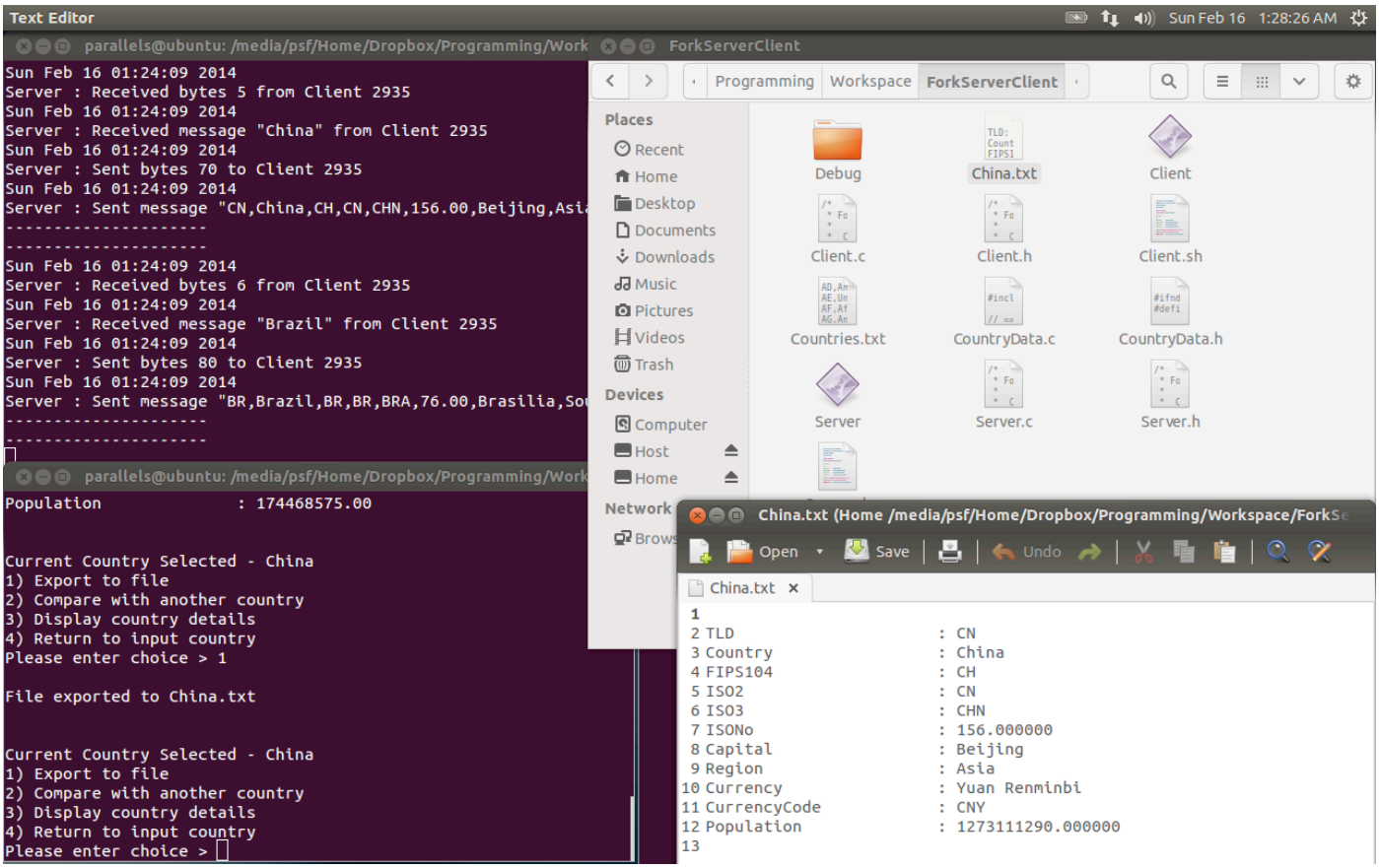
- Client has searched for a valid country data and was sent by the server



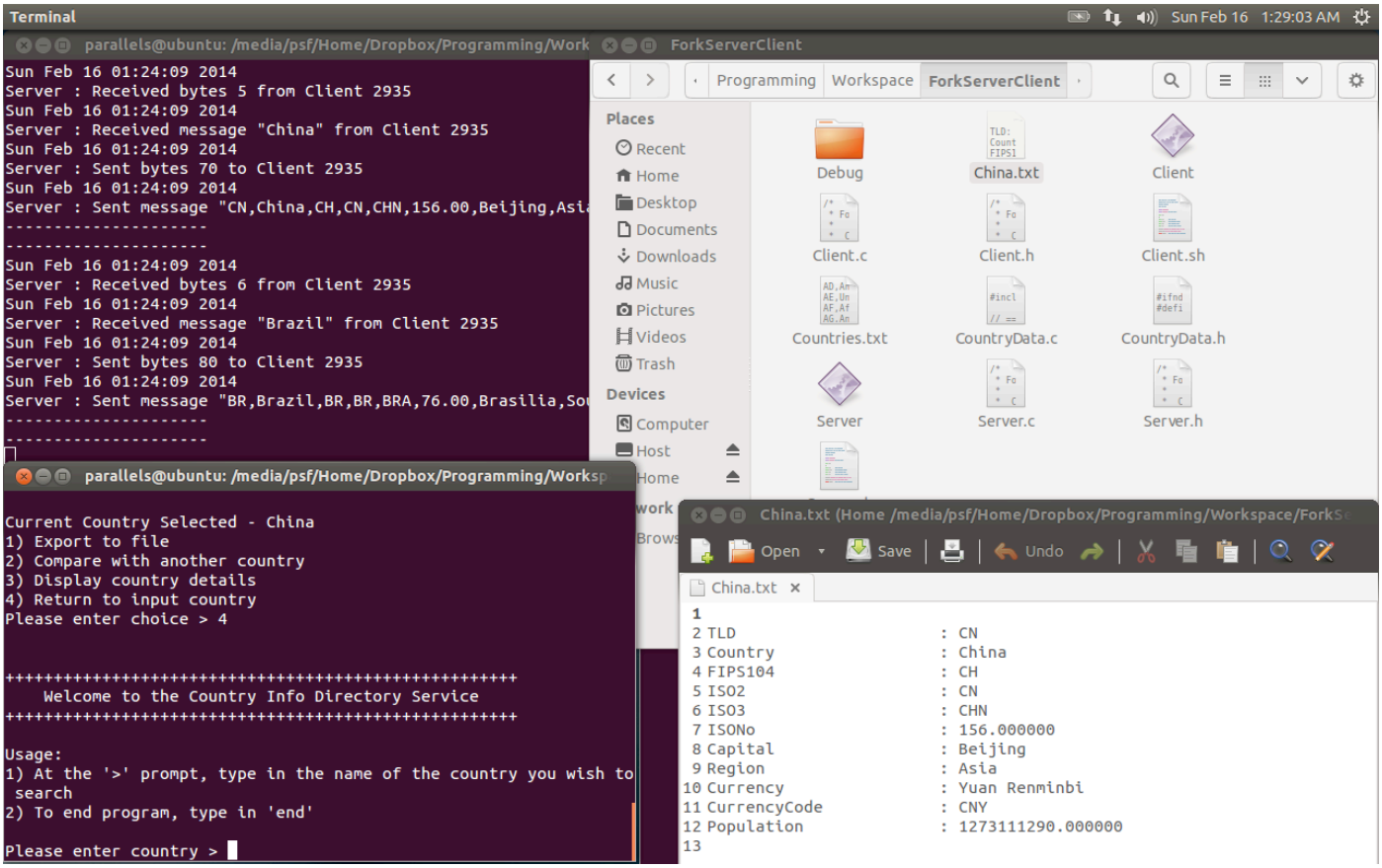
- Client picked option 3 from the menu, country details are shown



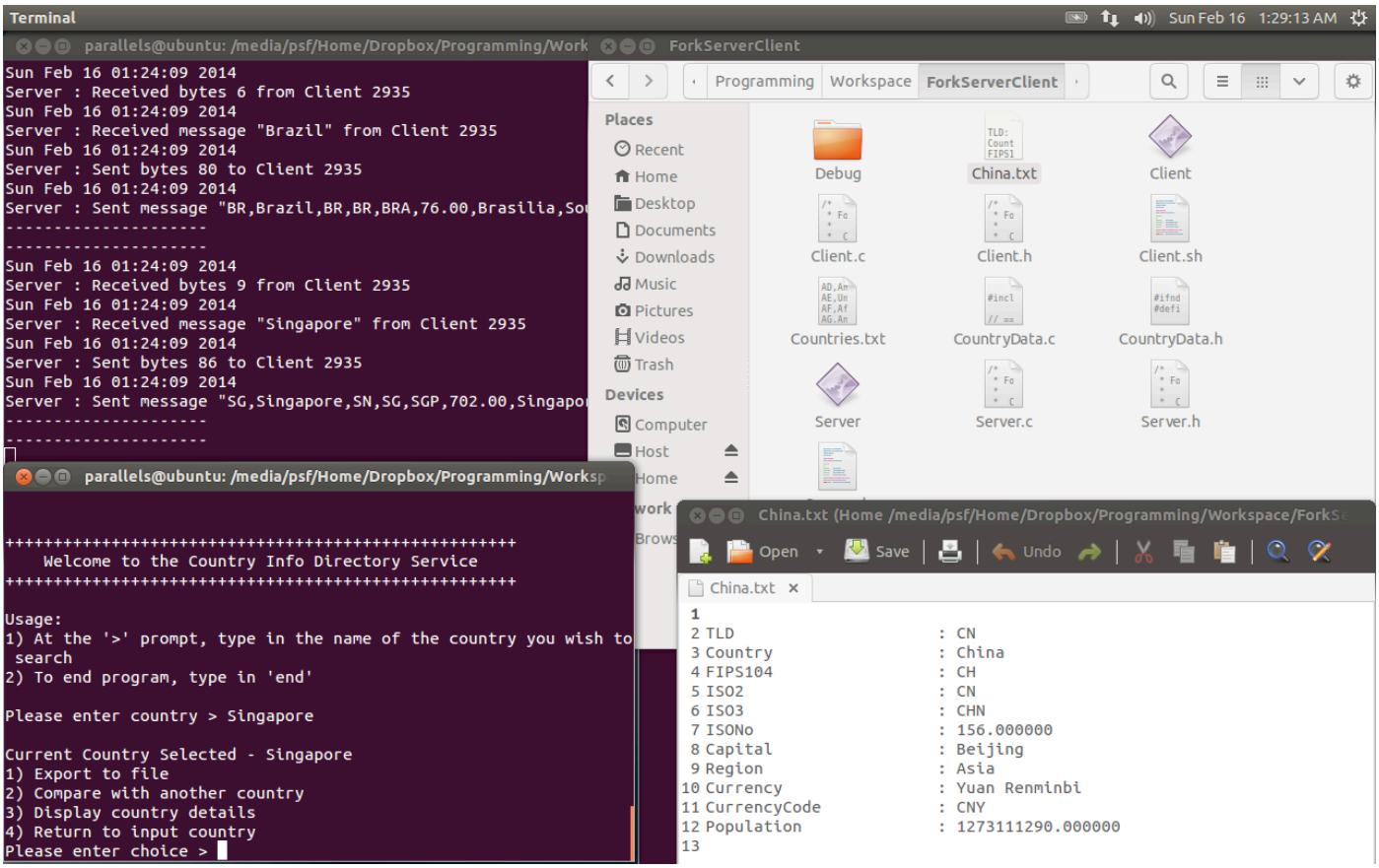
- Client has picked option 2, current country data is compared with another chosen country



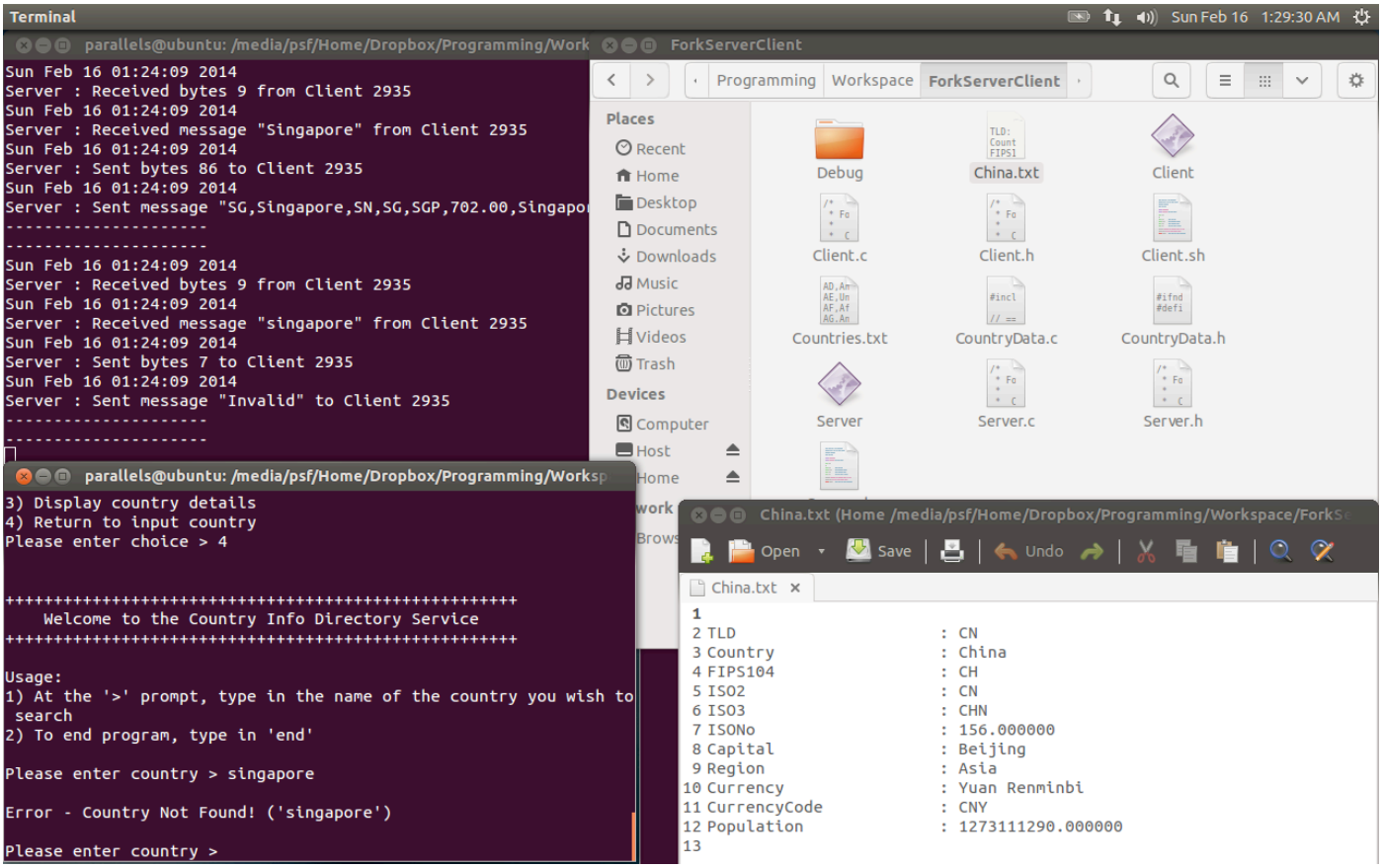
- Client has picked option 1 and has exported data to a text file



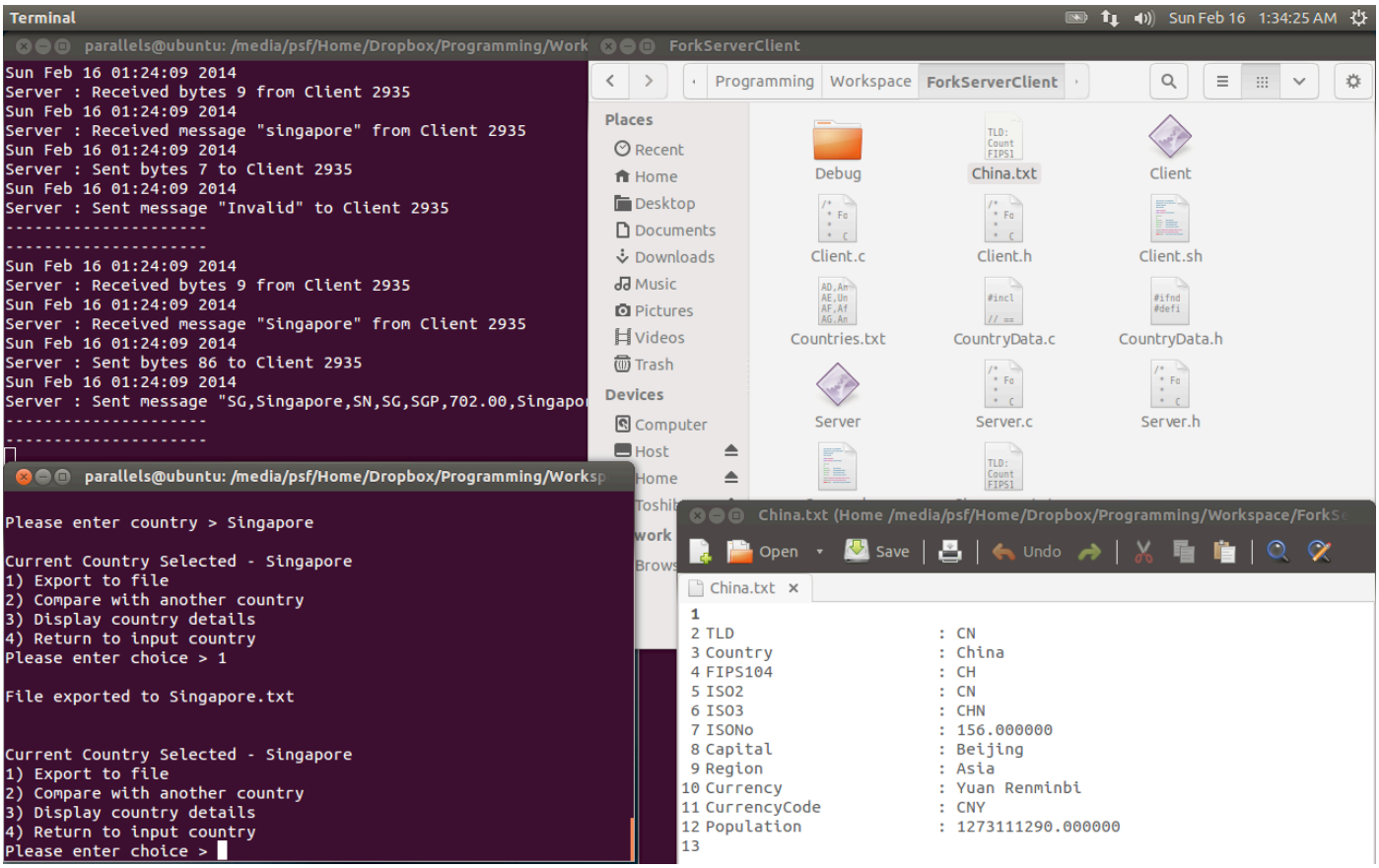
- Client picked option 4 and has returned to the country input prompt



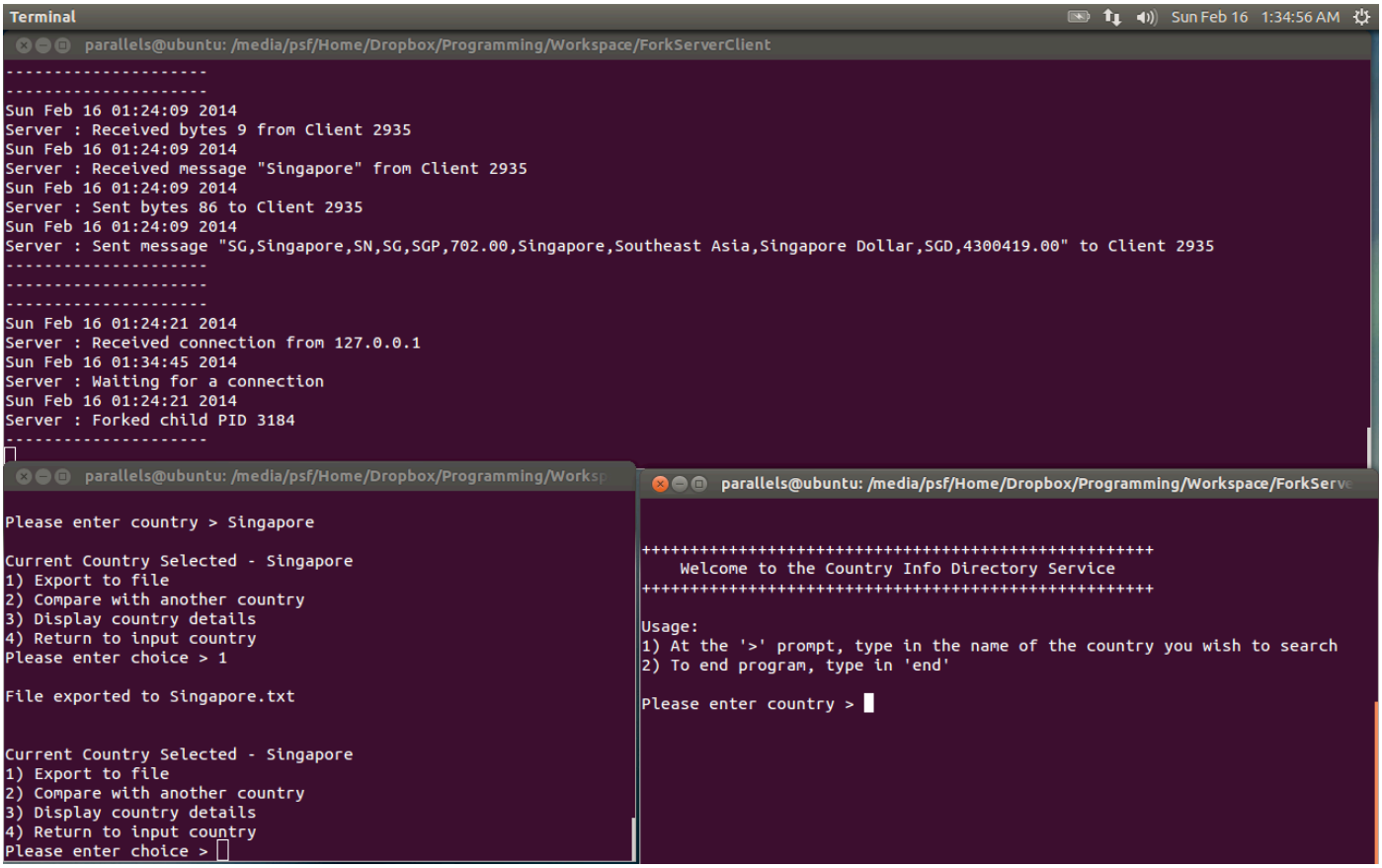
- Client has input another valid country



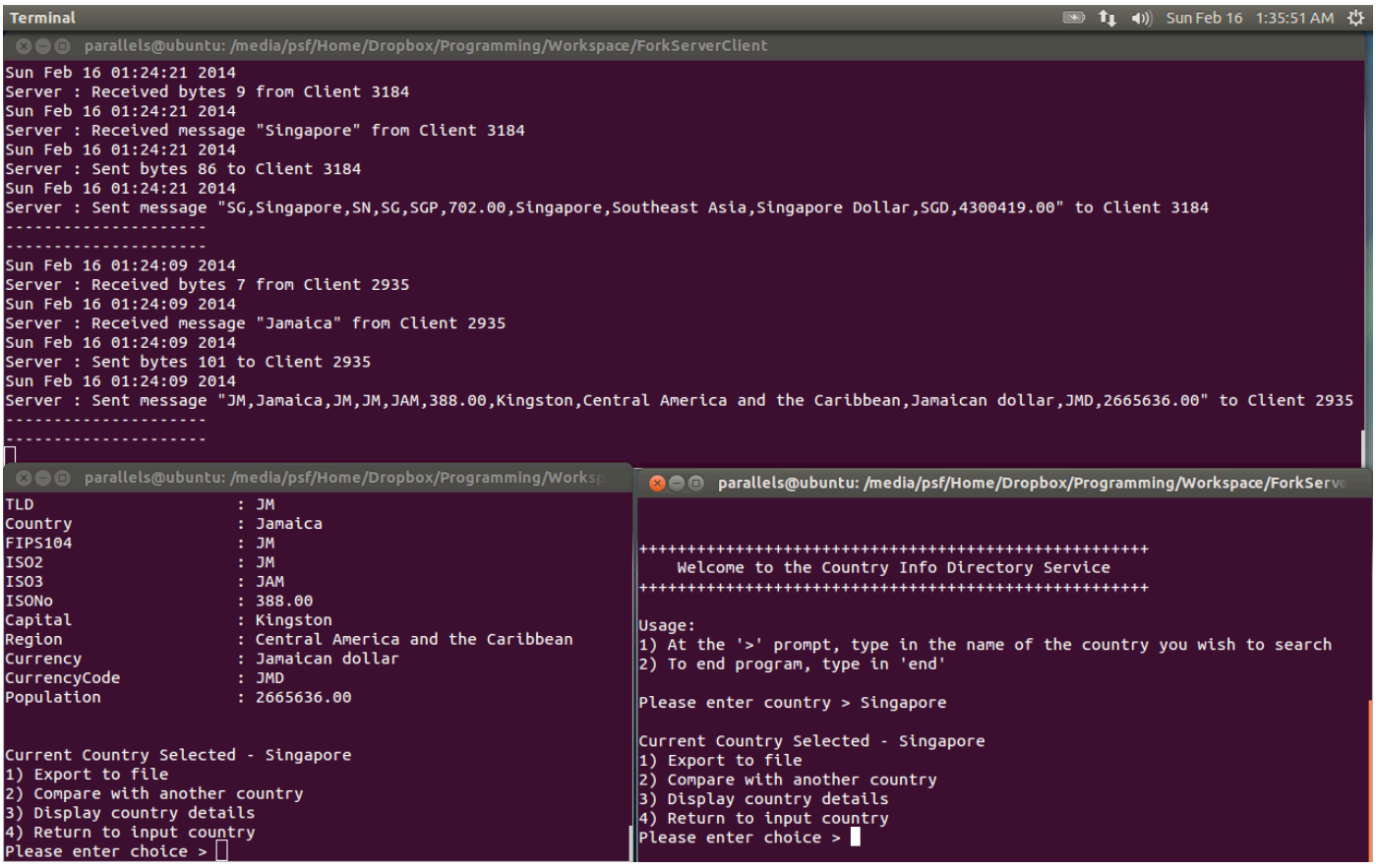
- Client went back to menu and input an invalid input



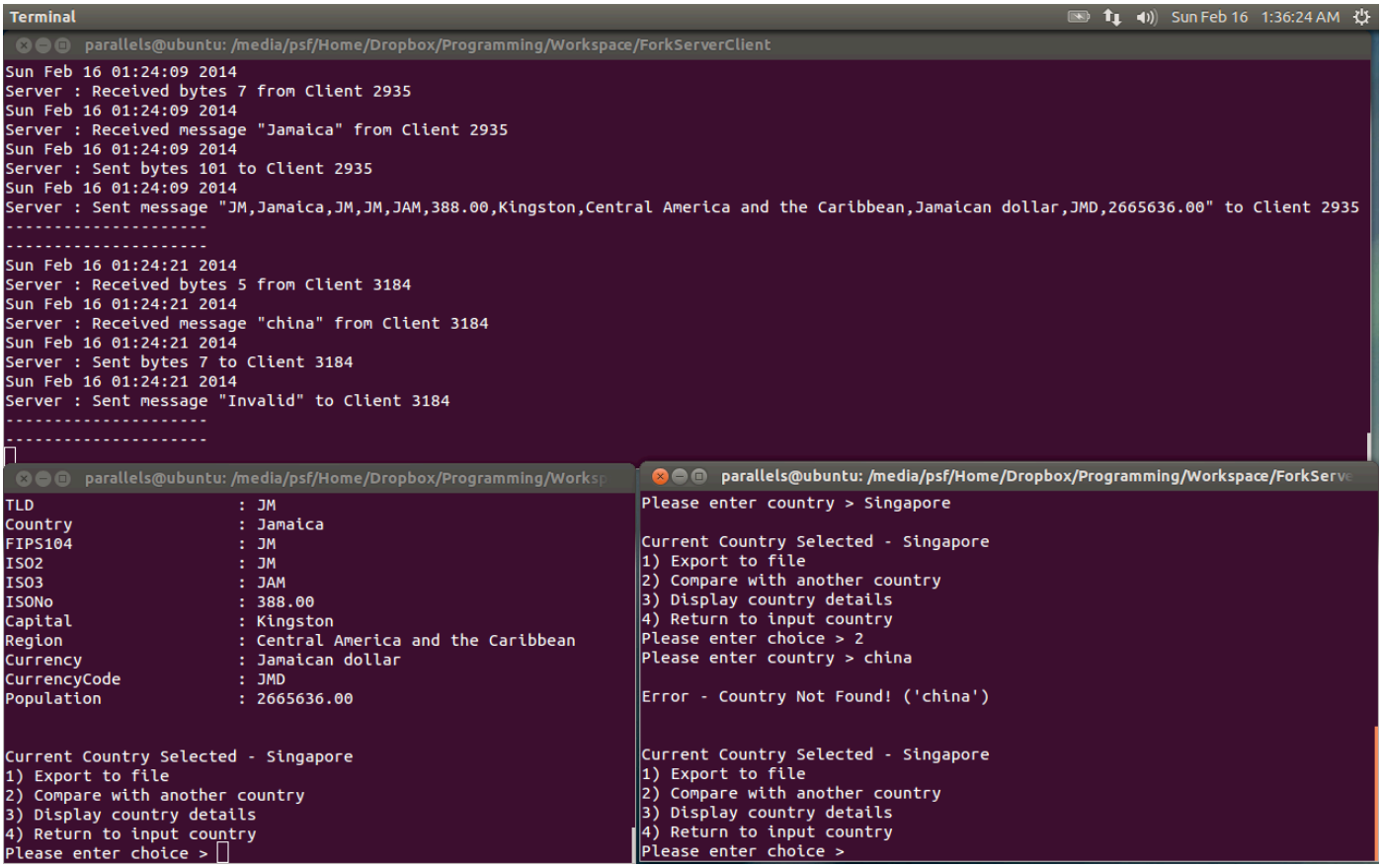
- Client exported newly searched data into another file



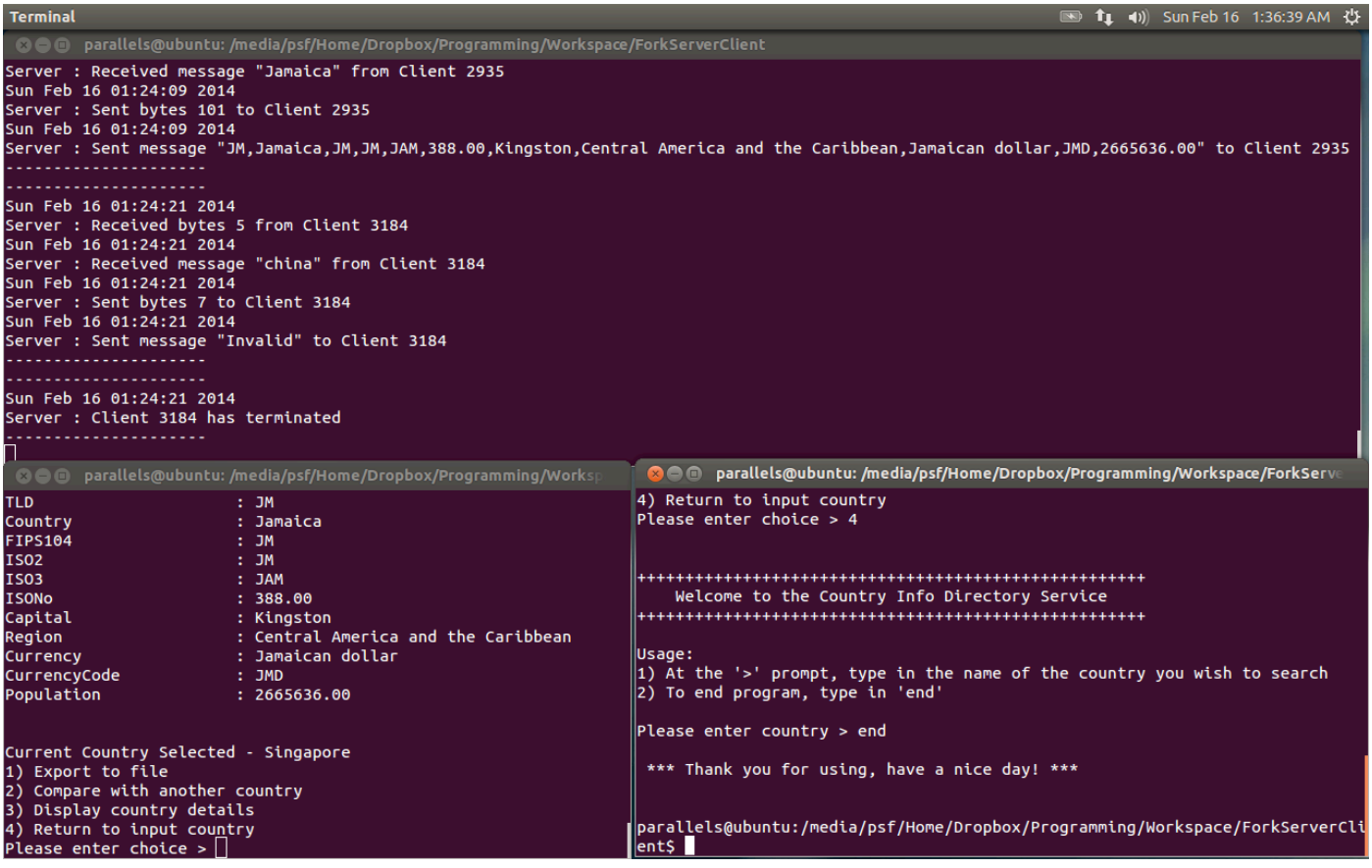
- Another client has connected to server



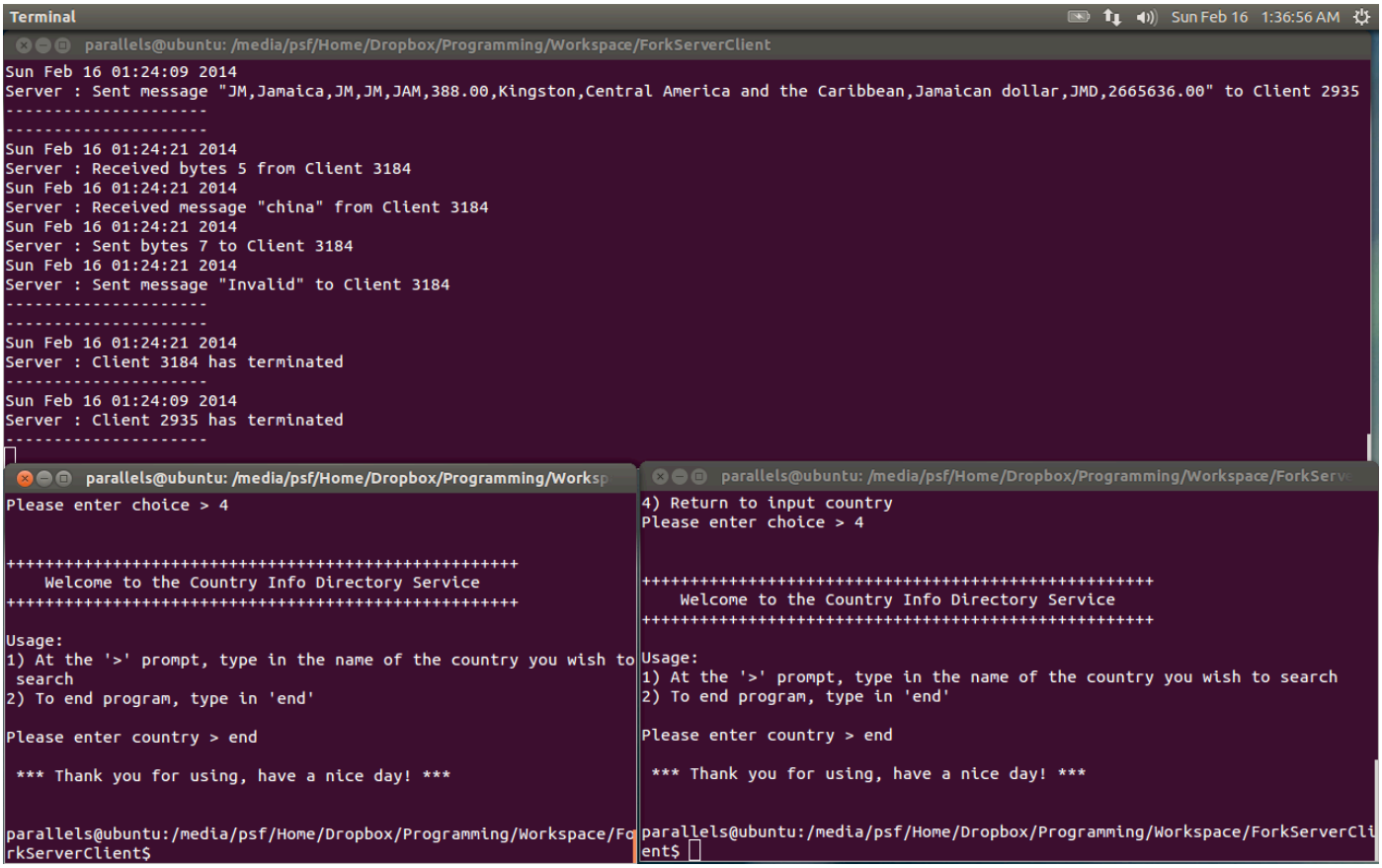
- Client has searched a valid country



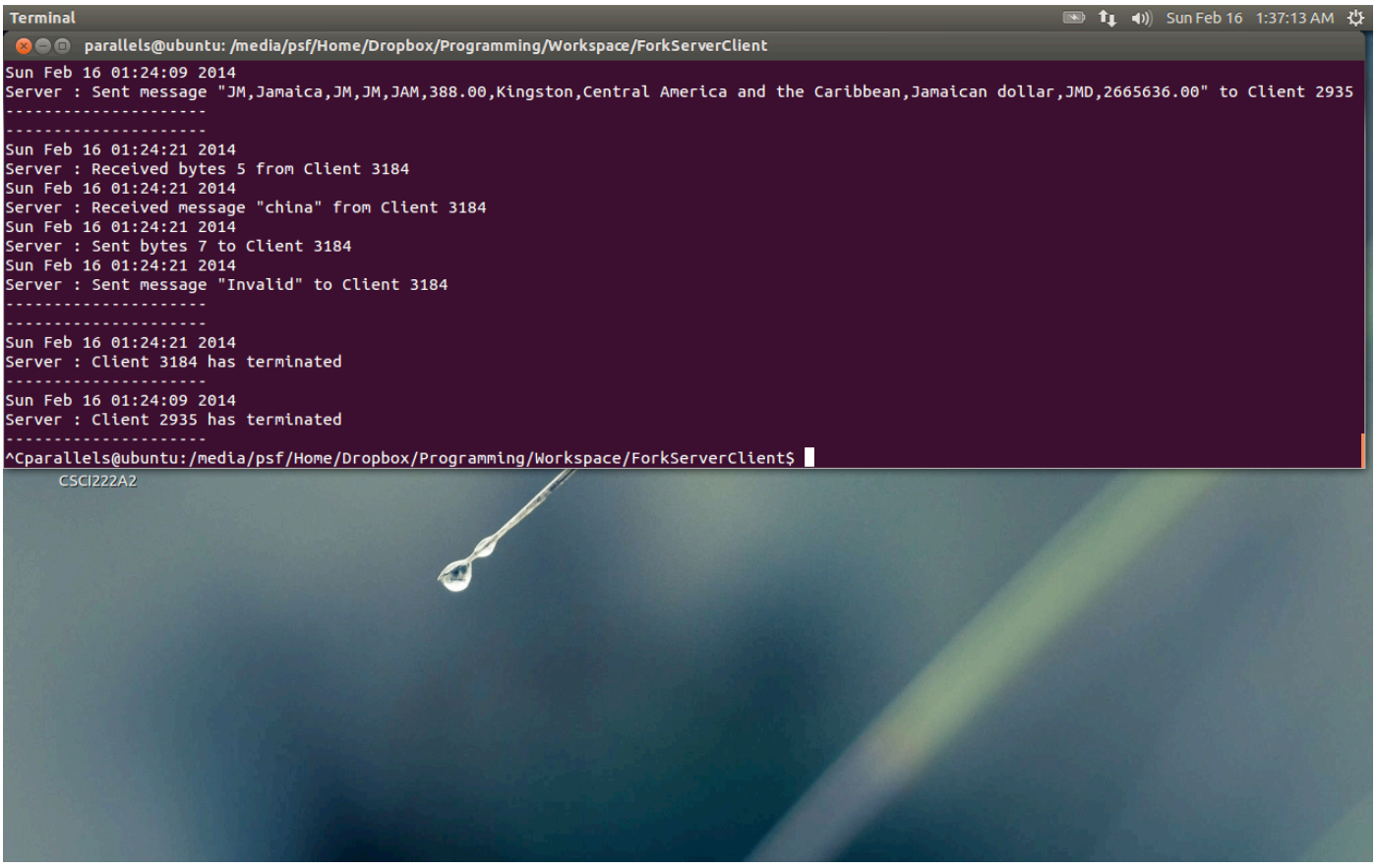
- New Client has searched invalid data and older client searched for another country (Jamaica)



- New Client has quit. It can be seen in the server the time the client program has terminated



- First client has terminated



- Server has been terminated

Summary of implementation of each module in your program

CountryData.h

****

```
void readData ();
      - Read data from file
char* get_line (char *s, size_t n, FILE *f);
      - Read a single line from file
CountryRecordType createCountryRecord (char* aLine);
      - Creation of country from a single line from the file
void displayRecordContent (CountryRecordType ctryRec);
      - Display a single record
void showAllRecords ();
      - Display all record by using a loop
int findCountryRecord (const char* countryName);
      - Search a record using country name
char* getCapital (const char* countryName);
      - Get a capital of a specific country
char* getCurrencyCode (const char* countryName);
      - Get currency code of a specific country
CountryRecordType getCountryData (const char* countryName);
      - Get a record using a country name
void getCountryDataString(char getString[],const char* countryName);
      - Concatenate all data of a record to a string
```

Server.h

****

```
int Connect();
      - Connection of server: Included setting up of socket and forking child clients
void* SocketHandler(void*,int);
      - Handles forked out child process
void ProcessReceivedData(char[]);
      - Process data that's received and data to be sent
```

CountryData.h

****

```
void displayInstructions();
      - Display banner with instructions
int startMessaging();
      - Create a loop for sending and receiving messaged
int Connect();
      - Connection of client to server: Setting up of socket
void getCountry();
      - Display input prompt
int storeData(char[]);
      - Stores data into a record
int sendData();
      - Send data using a socket
int recvData();
      - Expects to receive data from server
void exportData(CountryRecordType);
      - Export data into a file
void displayRecordContent();
      - Display a record data
void displayMenu();
      - Show submenu
void compareCountries();
      - Compare another country data with the selected country
```

Reflections on program development

- Assumptions made
  - CountryData.txt is the name of the file containing data
  - Multi-threading not required
  - Client and server are both ran in the same computer (IP 127.0.0.0 is fixed)
- Difficulties faced
  - Understanding the socket code
- What could have been done better
  -
- Possible enhancements in future
  - Client can edit data in stored in server
- What have I learnt
  - Socket connection between two application
  - Formatting and string manipulation in C