Translation Server Project Report

Authors: Rauf Ibishov, Rashad Mammadov

1. Introduction

The Translation Server Project implements a system for translating words between English and French. This paper details its development process, from its inception through today. The system has gone through three distinct versions, each version adding new functionality and improved design, done through inter-process communication means.

2. Workflow

2.1. **Dictionary Management**:

- The server initializes by loading dictionary files from a folder.

- It dynamically reloads translations if new files are detected.

- Translations are stored in memory for fast access.

2.2. **Client Requests**:

- Clients connect via TCP/IP sockets to send translation requests in the format `<direction:word>`.

- `direction` specifies the translation type:

- `SIGUSR1`: English-to-French.

- `SIGUSR2`: French-to-English.

- The server processes the request and returns the corresponding translation.

2.3. **Dynamic Reloading**:

- If a requested word is not found in memory, the server reloads dictionary files to include new translations.

2.4. **Multithreading**:

- A dedicated thread monitors the dictionary folder for updates.

- Another thread handles client requests to ensure real-time responsiveness.

3. Version 1: Basic Signal-Based Server and Client

3.1 Server

In Version 1, the server handles translation requests using signals (SIGUSR1 and SIGUSR2). It loads dictionary entries from files in the 'dictionary' folder and stores them in an array. The server responds to signals by selecting a random word and providing its translation.

3.2 Client

The client sends translation requests to the server by sending signals. The type of signal determines the translation direction (SIGUSR1 for English-to-French, SIGUSR2 for French-to-English). The client repeats this process for a predefined number of requests.

4. Version 2: IPC-Based Design with Shared Memory and Message Queues

4.1 Server

In version 2, we added shared memory and message queues for more efficient communication. The dictionary entries are read and stored in shared memory by the server. And it also watches all files inside the dictionary folder for changes and loads any new file into shared memory.

4.2 Client

In Version 2, the client sends a request to the server via an intermediate message queue. The translation direction and word to be translated are included in every request. The server then replies with the translation or an error message.

5. Version 3: Unified IPC-Based Server with Dynamic Updates

5.1 Server

The last version brings all the improvements made on Version 2, and adding dynamic update to dictionary. The server that processes the requests from clients, through shared memory and message queues to provide real-time translations. It watches the dictionary folder and will automatically assimilate any files placed there.

5.2 Client

The Version 3 client sends translation requests in a user-friendly format. It communicates with the server through a message queue and displays the responses to the user.

6. Program Workflow

The program workflow across all versions is as follows:

1. The server loads the dictionary from files in the 'dictionary' folder.

2. The client sends a translation request to the server.

3. The server processes the request, retrieves the translation, and responds to the client.

4. In later versions, the server monitors the dictionary folder for updates and dynamically reloads the data.

7. Conclusion

The Translation Server Project is a nice example of inter-process communication mechanisms at work. Three versions demonstrate gradual refinements in design, power and utility. This project showcases the possibility of having real-time updates and