COIS3380 Mini-Project Stage 2 – Pokemon Server

Prerequisites

Please review the lectures up to and including Week 10.

Download the csv file containing Pokemon descriptions.

Learning Objectives

- Convert a set of requirements into the program design and implement running code
- Make your stage 1 and 2 code deployable and usable by different actors in a client-server arrangement.

Learning Outcomes

1. You will learn how to develop client-server applications using sockets.
2. You will learn how to create a simple C build using Makefile.

Client-Server Pokemon Query Program

The goal of this assignment is to develop a computer program that helps explore the properties of Pokemon:

> *You are an avid gamer. You really like Pokemon games and trading cards. In order to improve your Pokemon gaming abilities you need to explore the properties of different Pokemon. A CSV file listing all Pokemon and their properties is available in the public domain. You would like to explore the file to learn more about different Pokemon and their properties. To do this you want to write a program that will allow you to look up Pokemon by name and store their properties in a separate file. You travel and are not able to always have the Pokemon file with you. You would still like to be able to explore Pokemon properties from a remote location by starting a light-weight client application that can connect to the server that can perform queries on the Pokemon file.*

Legal Disclaimer

Pokemon and all related images, trademarks and copyrights are property of their rightful owners. This assignment is intended as a fun learning exercise, and is not intended to infringe on any of these copyrights and trademarks. Please consult this site for legal obligations regarding Pokemon: https://www.pokemon.com/us/

Functional Requirements

Use Case 1 – Administrator starts the Pokemon Property Server (PPS).

Actor: Administrator (for the assignment this is the same person as the Gamer - you)
Steps:

1. The Administrator starts the PPS program using a Linux terminal.
2. The PPS program prompts the Administrator to enter the name of the file containing the Pokemon descriptions.
3. The Administrator enters the name (or full directory path to, but we can assume the file is in the same directory) of the pokemon.csv file
    a. **Error flow:**
        i. If the file is not found the query program displays the message: “Pokemon file is not found. Please enter the name of the file again.”
        ii. The query program prompts the Administrator to enter the name of the file again, or to exit the program.
4. The PPS starts listening on the pre-configured IP address and port (for this assignment both PQC and PPS will run on the same machine. The PPS should listen on localhost (127.0.0.1) port 80. The IP and port can be hard-coded for both the PPS and PQC).

Use Case 2 – Gamer starts the Pokemon Query Client (PQC) and connects to the Pokemon Property Server (PPS).

Actor: Gamer
Steps:

5. The Gamer starts the PQC.
6. The PQC program establishes connection to the PPS.
    a. **Error flow:**
        i. If PQC is not able to establish a socket connection to the PPS, it displays an error message to the Gamer (“Unable to establish connection to the PPS!”)
        ii. The PQC exits.
7. The PQC program displays a menu with the following options:
    a. Type search
    b. Save results

        c. Exit the program

Use Case 3 – Gamer searches for Pokemon by type (Type 1) using the PQC.

Actor: Gamer
Steps:

1. The Gamer selects option 1 on the PQC menu – Type search.
2. The PQC program sends the query to the PPS, which searches all Pokemon data in pokemon.csv, finds those Pokemon records where Type 1 matches the Gamer's input, and sends them back to the PQC.
3. The PQC program stores the accumulated query results in its memory.

Use Case 4 – Gamer saves the accumulated query results using the PQC.

Actor: Gamer
Steps:

1. The Gamer selects option 2 from the PQC menu – Save results.
2. The query program prompts the Gamer to enter the name of file which will contain the query results accumulated to this point.
3. The query program saves the results of all completed queries in this file.
    a. **Error flow:**
        i. If the query program is not able to create the new file it displays the message: "Unable to create the new file. Please enter the name of the file again."
        ii. The query program prompts the Gamer to either enter the name of the file again.
    b. The query program does not include any data generated by queries that are still in progress.
    c. The save operation runs in the background, and the query program immediately displays the menu described in Use Case 1.

Use Case 5 – Gamer exits the PQC program.

Actor: Gamer
Steps:

1. The Gamer selects option 3 from the menu – Exit the PQC.
2. The PQC program ignores any queries that are currently still in progress on the PPS.
3. The PQC program displays the total number of queries completed successfully during the session.
4. The PQC program displays the names of new files created during the session.
5. The PQC program exits.

Non-Functional Requirements

Computer programs need to meet not-only the requirements of people who use them, but also of people who need to maintain, operate, extend, and debug them. These types of requirements are called Non-Functional Requirements (NFRs). Typically, the non-functional requirements focus on the following:

- Performance
- Maintainability
- Security
- Availability
- Extensibility
- Others…

For this assignment we will focus on the following NFRs.

NFR1 – Performance

The query program must remain responsive to the user input regardless of the number of queries that have been submitted by the Gamer, while queries are executing, and while writing the results to disk (HINT: you accomplished this via multi-threading in Stage 2).

NFR2 – Maintainability

The PQC and the PPS must be implemented using separate C and header files so that different programmers could work on the client and server portions in parallel. A custom Makefile is needed to facilitate builds.

Deliverables

This assignment will have two deliverables:

1. **Readme File.**
    a. Submitted in text format.
    b. It should explain how to build and run your program.
    c. It should also explain the key program design and implementation choices to supplement the comments in your code.
2. **Makefile**
    a. The Makefile should build all of your code for both the client and server portions of the applications using a single make command.
3. **C code.**
    a. C and header files that implement the program design.
    b. For full marks the code should:
        i. Run without errors.

    ii. Implement the functional and non-functional above - correctly.
    iii. Should include comments that explain key implementation choices and follow the ANSI C style.

4. **Guidellines**.
    a. ***Aim to complete coding for this stage within 4 days.***
    b. If you work as a team, try working on your code together (pair programming).
    c. Submit your code and readme when ready.
    d. Organize your code nicely into functions.
    e. Different team members can work in parallel on the client (PQS) and server (PPS) components.
    f. Only the final deliverable will be graded, but if you are not able to complete all stages they will be able to get partial marks for code that you submitted.
    g. If you had trouble with Stages 1 and 2 try to get that code working prior to stating on Stage 3. Make sure that your latest submitted code, is working code!