INX 365 Assignment 2
Distributed Communication

Lukas Elsner
n9224262
Queensland University of Technology
September 30, 2014

The task was to develop a client/server application for professional dietitians using C and BSD sockets. The server should load a csv-file with food and their nutrition values. Furthermore, it must handle multiple client sessions at once. Every client can query food by search terms and add new food to the list. This document covers the implementation report, as well as occurred problems while implementing the application.

Contents

1 Statement of Completeness

All three tasks has been implemented as requested. Some minor additional changes were made. This report will cover the implementation details, as well as the additional changes.

1.1 Basic Functionality

The following basic functionality was implemented:

- Server automatically loads calories.csv from the current path. If the csv-file is in the same directory as the server binary, it must be run from this directory.
- The server's listening port can be changed with help of a command line parameter
- The server initiates a clean shutdown when it receives the SIGINT signal.
- The server can handle 10 client connections in parallel. A producer/consumer pattern was used, as Task 2 suggested this.
- Food can be added from a client by pressing 'a' and then following the on screen instructions to supply the needed attributes.
- The client accepts two command line parameters. The server's IP-address and its port.
- When the user enters a food name, the server processes the request and sends the search results back to the client.
- If a search query results in no findings, an appropriate error message is printed on the client side.
- The client can be shut down by pressing 'q'

In addition to that, the following extras were implemented:

- Running the client or server with $-h$ parameter will print the usage help.
- The client connects to 127.0.0.1 on port 12345 when started without parameters.
- The client connects to 127.0.0.1 on given port with only one parameter.

1.2 Process Synchronisation and Coordination

1.2.1 Connection handling

To be able to handle multiple connections, a thread pool is created. Ten threads are consuming the clients produced by the server thread. A consumer/producer pattern has been used with help of mutexed, semaphores and a rotating array for the data which has to be accessed by multiple threads.

1.2.2 Data handling

All food is stored in a linked list. Multiple threads can read the list at any time, but only one thread can write. This is implemented by the Reader/Writer pattern from lecture 5.

1.3 Non-Functionality

Valgrind confirmed that there are no possible memory leaks, even after a long runtime of both the client and the server.
Also, the code is well commented and a doxygen reference manual was created.
There are no known issues regarding to unexpected behavior, such as crashes.