CSCI 5531 – Spring 2016

Advanced Operating Systems

Team Project

**Team 6**

Amey Parab

Abilash Reddy Kommareddy

Ha Vu

Table of Contents

1. **Project Overview1**
2. **Challenges4**
3. **Flow of functionality4**
4. **Flow Diagram**5
5. **Conclusion**6
6. **Project Overview**

This project implements a simplified character-based email and calendar service. The program will support multiple clients. They can send and receive email from each client. Furthermore, this program enables users to upload and download attachments in text based format. In addition to email service, Calendar service allows users to set up appointments with each other.

The program is written in the Client/Server architecture. The environment is UNIX, and the language is C. The server is a concurrent server in which each connection is handled by a child process. All processes communicate through shared memory and semaphore.

The shared memory is a nest Struct array which contains active user detail information such as user name, password, emails, calendars… This structure organizes data in a logical manner similar to the data encapsulation concept of Object Oriented Programming where each element in the array acts as an object.

The client side contains 2 threads. The first thread sends requests to check if there is any new email or calendar every 5 seconds. The second thread enables user to interact with the server. Users use this thread to check or send email, schedule or modify calendar, and so on…

1. **Challenge:**

**Design Challenge** The program must support multiple clients. Concurrent server is the best model that can handle this requirement. The server will fork a child process to handle each connection. Since there are multiple clients, and each client will have 2 connections to the server. If all data, such as email and calendar were maintained and accessed through text files, the performance of server will be slow due to multiple IO operations on each request. To improve performance, all data will be loaded into a nested struc array. The server will use this array to handle all request and only write it back to file only when users log out or the server is shutdown.

The issue is how the server can distinguish which request is from which client. To address this issue, the server maintains a struct array of all active users (active users are those who are currently logged in). Each request from client will contain a user ID. Server will use this user ID to identify to whom this request belongs to. Each request also contains request command which tells the server what the request is for e.i CMD\_CHECKEMAIL, CMD\_SENDEMAIL….

The next challenge is how information can be shared between different users. Shared memory and lock were used to handle this task. The nested struct array was used as shared memory. Each process can access this shared memory at a time.

On the client side, the user must be notified if they have new emails or appointment when they log in. This must be done automatically. To accomplish this task, client side uses two threads each thread has its own connection. One connection is from the background thread which is transparent to user. The background thread will send request to server every 5 seconds for new email and calendar. The other connection is for the foreground thread which handles interaction between client and server.

**Implementation Challenge**: Since all data is loaded into array, which might cause over load the memory. To minimize this risk, only active users are kept in memory. Once the user logs out, all of their data will be written to file to free up memory.

Also, because there are multiple processes, and each process has its own port. The server needs to close all open ports when user logs out or the server is shutdown.

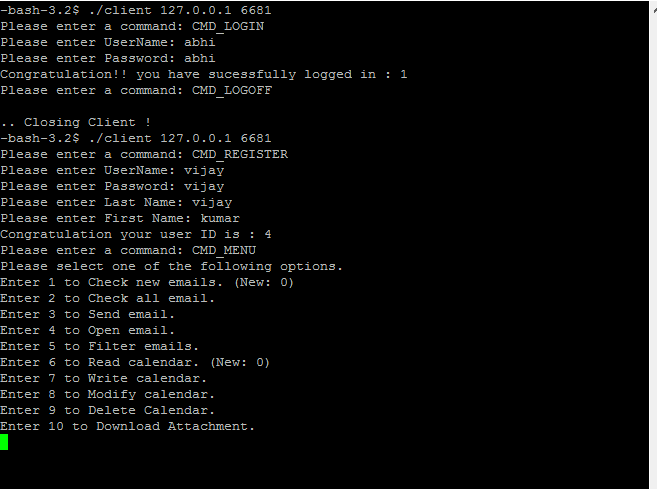
Besides all the technical challenges, another challenge that our team faced is that each of us has different scheduled. It has been difficult for us to set up a meeting that all could attend. We had to meet on the weekend to work on this project to meet weekly objectives.

3. **FLOW OF FUNCTIONALITY**

In this email/calendar system, the client is designed as a multithreaded program (2 threads) one thread which interacts with client and second is background thread which runs for every second to get new mail or calendar events. On the other hand the server is designed to work as a multiprocessing program that forks a new child process for every new incoming client request.

Initially the program begins with establishing the connection between the client and the server using the sockets.

User will be asked to enter command, if its new user and he/she has enter CMD\_REGISTER and detailed information about user. It is collected and stored in the in the configuration file of the system. The details include, First name, Last name, Email Id and Password and stored in users.txt file. If user has already an account then user has enter CMD\_LOGIN command and enter email Id and password.

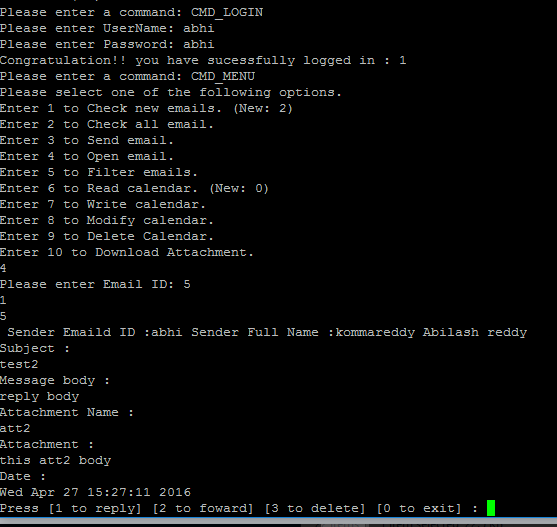


Once user authenticated, User will be asked to enter command CMD\_MENU or CMD\_LOGOFF.

CMD\_LISTUSERS, will display list of registered users

If user enters CMD\_MENU, below list of email and calendar services are displayed.

1. Check new email : Displays list of unread emails(email number, subject, date)
2. Check all email : Displays list of all emails (email number, subject, date)
3. Send email: User has to enter receiver email id, subject, body, attachment name (if required), and attachment body (if required). If user enters valid information User will get notified whether mail has been delivered successfully or not.
4. Open email: User will be asked to enter email number, then it displays all email information. Displays below options.
   * 1. Reply email
     2. Forward email
     3. Delete
5. Filter email: User will be asked to filer email by sender name or subject or date. It displays filtered emails.



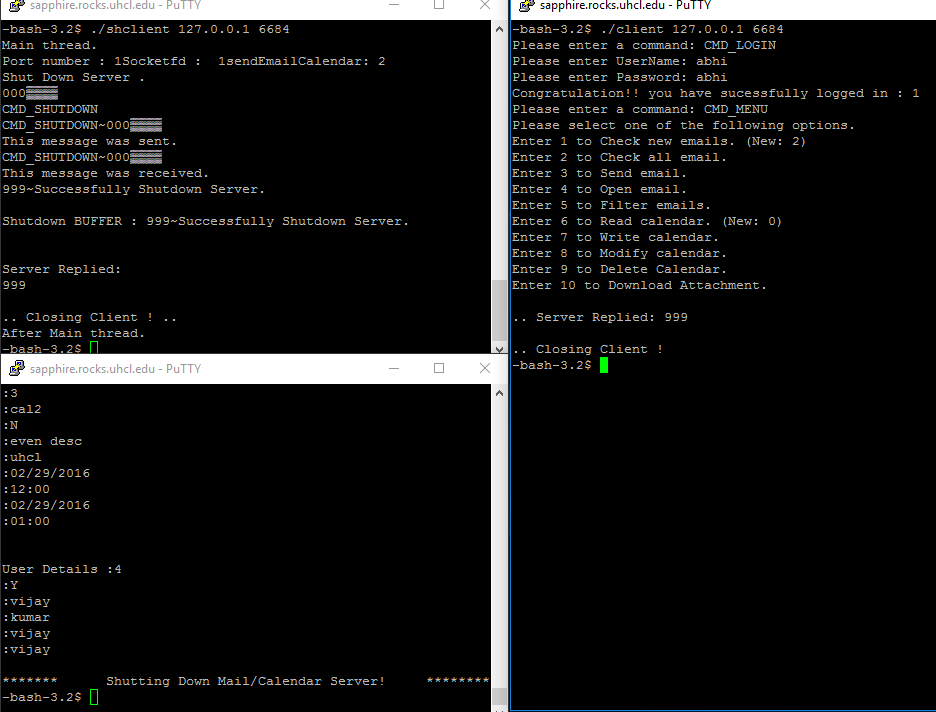
1. Read calendar: Displays list of calendar events of the user.
2. Write calendar: User will asked to enter recipient Id’s, Subject , Event description , Location, Start date, Start hour, End Date, End date. If user enters all valid information, will get notified whether event created successfully or not.
3. Modify calendar: User will be asked to enter event Id of which he/she wants modify, then user has to enter updated calendar information.
4. Delete calendar : User will be asked to enter event id and user will get notified

Whether deleted successfully or not.

1. Download attachments: User will be asked to enter emaild id or calendar event id, text file will contents will be downloaded.

Once User is done with all the Tasks User Can User CMD\_LOGOFF command to successfully save his all new changes persistently on the server and close the console.

We Have Also created ShutDownClient to gracefully shutdown the server by sending CMD\_SHUTDOWN internally. This program will be run separate client. This program will write whole data from the shared memory into files on server and Notify all clients that are connected that server has been closed and close the clients with the server

.

IV. FLOW DIAGRAM

C:\Users\ABILASH\Desktop\Master's\AOS\flow_diagram.png

**V.CONCLUSION**

In this project, we have developed an email system that works on the Linux platform and is coded by the C programming language. It has included all the elements of C programming in it and has been a good experience working on. The client is a multithreaded program where as the server is a multi-processed program .The client part and the server part of the program communicate with each other through the sockets. The file systems are used in other to store the user information and the data relevant to each user. Thus, this project has given us a good programming experience on almost all the basic entities of the C programming on Linux environment.