Mail Runners

Design Document:

Client Side Interface

The portion of the application directly accessed by the user will be a separately compiled client which will allow users to login in and interact with the server. The client sideinterface will generate requests based on user input and forward them to the server for processing. Any output from the server will be formatted and displayed in the interface. When the client is executed it will prompt the user for a username and password. The credentials are first checked against predefined constraints such as a username more than 3 and less than 20 characters. The password must be more than 6 characters and less than 20 containing alpha and numeric characters. If these constraints are met, the client application will request a connection to the server passing the username and password. If the client receives confirmation of a connection, it proceeds to a screen allowing the user to choose between File and Mail. If the connection failed due to an incorrect username/password the user will be prompted again.

If the user selects File, the screen will change to a list view of the top most level of folders. This top most level will be the public portion of the file structure. From within this list view a user can click on a folder to drill down and update the folder to show the contents of the selected folder. If the user does not have permission to the selected folder it will remain at the current level and display an error message. The user also has the option to traverse up the file tree to the parent directory. If the user selects a file within the list view they will be prompted for a destination and the file will be downloaded. The file is transferred from the server into the internal cache of the client, and then moved from cache to the destination provided by the user. There will also be an option on the list view to upload a file into the current directory. The user is prompted for a source and the file is first copied into the cache, and transferred through the communication channel to the server.

If the user selects Mail they will be presented with three options, Receive New Messages, View Inbox, and Send Mail. Choosing Receive New Messages will send a request to the server and return either the number of new messages received, or a message indicating that there are no new messages. Choosing View Inbox will show a screen similar to the folder list view except it will show the mail messages including who sent it, and when it was received. If the user chooses to send a new message, they are first asked to select a recipient from a list of users. Once a recipient is selected the user will be prompted for a source. The message will be uploaded to the server as if it were a file, with the exception of adding additional information such as sender and recipient etc.

At any time a user will be able to go back, which will return to the previous menu, or logout. Input from the user will be in the form of mouse clicks, or in the case source and destination prompts, text input. Logging out will send a request to the server to sever the current connection.

Communication Manager

The Communication controller is responsible for setting up a communication pipeline between the Client and the Server. It will technically have parts in the client and server sides of the application. The communication controller is invoked by the client interface at the start of the application and is responded to by the communication controller component residing on the server.

Once a username and password are entered, the client end requests a connection. The server listens for the request and passes control to the security manager to verify the login credentials provided by the client.

If the credentials check out, the communication controller establishes a connection with the server else it returns an error message back to the client. It is also responsible for closing the connection when the user logs out.

The communication controller achieves this functionality by using an application programming interface (API) for inter process communication known as a Socket.

Sockets are a protocol independent method of creating a connection between two processes. Each Socket has one or more protocols associated with it, for example TCP/IP (virtual circuits) and UDP (datagram).

Working of the Communication Controller

For a better understanding of how the communication controller works we can individually look at the working of its two components, namely the client and the server.

The Client:

- At first the client creates a socket and attempts to connect to the server.
  Methods Involved:
    - Socket (domain name, type, protocol)
        - Creates a socket to set up a communication channel
    - Bind (socket id, address structure, length)
        - Associates a socket id with the address structure so other processes can connect to it
- The client sends the username and the password to the security manager residing on the server for authentication.
- Once the security manager grants access to the user the communication controller sets up the communication pipeline. If the client authentication process fails than an error message is returned back to the client.
- Now that the pipeline has been established, the client and server can actively exchange information.
  Methods Involved:

- Send (socket id, data, length, flag)
  - Sends data using the socket id to identify the server and uses the flag for acknowledgement
- Recv (socket id, data, length, flag)
  - Receives data using the socket id to identify the server and uses the flag for acknowledgement

- Once exchange of data has taken place it is important to end the sending and receiving of data and close and release the kernel data structures.

Methods Involved:

- Close (socket descriptor)
  - Implements one way close of a socket descriptor
- Shutdown (socket id, how)
  - Disables sending or receiving of data

The Server:

1) At first the server creates a socket.

Methods Involved:

- Socket (domain name, type, protocol)
  - Creates a socket to set up a communication channel
- Bind (socket id, address structure, length)
  - Associates a socket id with the address structure so other processes can connect to it

2) The server now listens and waits for connection requests from clients.
Methods Involved:

- Listen (socket id, size)
  - Where size is the number of pending socket requests allowed

3) Once the server receives a connection request from a client it calls the security manager to verify the credentials given by the user.
4) If the client clears the authentication process the server continues to go ahead and setup a communication channel between the client and server.
Methods Involved:

- Accept (socket id, address structure, length)
  - Returns the socket id and address of the client connecting to the socket

If the client fails to pass the authentication process an error message is returned back to it

5) Once the communication channel has been setup, the server and client can exchange data. Methods Involved:

    - Send (socket id, data, length, flag)
        - Sends data using the socket id to identify the server and uses the flag for acknowledgement
    - Recv (socket id, data, length, flag)
        - Receives data using the socket id to identify the server and uses the flag for acknowledgement

6) Once exchange of data has taken place it is important to end the sending and receiving of data and close and release the kernel data structures.

    Methods Involved:

    - Close (socket descriptor)
        - Implements one way close of a socket descriptor
    - Shutdown (socket id, how)
        - Disables sending or receiving of data

Security Manager

For the implementation and design of our project, we will be considering the Security Manager as a separate component. The Security Manager will be responsible for maintaining usernames, passwords and relationships as to what folders a given user has access to. It will regularly be referenced by the File Manager when attempting to access a folder to ensure that the current user has access to the folder. For our purposes it is assumed that if a user has permission to access a folder, it also has permission to access all of the files within it.

1) Called to validate a username and password when the Communication Manager attempts to create a connection.

   Methods Involved:

   - validateCredential(username, password)
     - Finds the username in the multidimensional array contained within the Security Manager, after finding the username in the array, it pulls the password and compares it to the password passed to the function. If the username cannot be found or the password does not match the method will fail

2) Whenever the File Manager attempts to view a folder, the path and username are first passed to the Security Manager to ensure that the user has permission to access the folder.

   Methods Involved:

   - verifyUserPermission(username, path)
     - Finds the username in the hash table contained within the Security Manager. Attempts to locate the path provided in the table to ensure that the user has permission to that folder.

   If the user is a Super User, they will have permission to access all of the folders on the server.

File Manager

The major responsibilities of the file manager will be as follows:

1) List:
   - Initially it will take a folder path and a username of the person who is requesting the list action. It can start from the public or the private directory. For both cases, first it will verify with the security manager that the user has access to the selected folder/file.
   - The public folder is created for the public where each user can access to the folders or files belong to that folder. Whereas private foldersare created for the specific users. Each private folder will have one index folder.Attempting to navigate to another user's private folder/file (when not a super user) will result in an error message.
   - The traversal mechanism between the public and the private folder will be much like LS/CD in a UNIX system, a user can select a subfolders/files or return to the parent folder/file.

2) Upload:
   - When a file comes to server through the communication pipeline the server will first store the file to its cache and then moved that file to the right directory based on the path provided. For every case the user's permission level should be checked before completing this upload request. A positive acknowledgement will be sent to the client if the permission level succeeds or a negative acknowledgement will take place. For both cases it will confirm the upload operation of our file manager.

3) Download:
   - In this mechanism the server will get the username and a specific path for a file to be downloaded.
   - First it will verify the user access to the specific folder/file and based on the access level permission it will send the file through the communication pipeline. If the permission is not granted then the server will send a negative acknowledgement to the client and will be activated upon next call.
   - Moreover, if the permission is granted then a positive acknowledgement from the client will be sent to the server just to confirm a successful download transaction.

4) Info:
   - Some of the information will be kept related to file or folder such as the size, the time created and the time for last modification.
   - This information will be saved by creating a separate hash table into the server (though each folder or file would have their unique ID) and will be available upon client's request.

5) Move:
   - The File Manager will be given source and destination paths along with the proper access authority. The file manager will then copy of the source file at the destination path, when the file has successfully been created, the original will be deleted.

Mail Manager

The Mail manager will have two levels, the Mail Depository (level 1) and user inboxes (level 2). This folder hierarchy will exist in an isolated folder tree away from the user folders.

1) Receive Mail:
    - The Mail Manager will receive a request with a given username. It will then scan through the Mail Depository for any messages with the given username as a recipient. If it finds a message, it will invoke the File Manager to move the message from the Depository to the user's Inbox.
2) View Inbox:
    - Takes a username and invoked the File Manager List function to return the contents of the user's inbox. If the user is a Super User they will be given access to the entire mail system, not just their own inbox.
3) Send Mail:
    - After the user has finished forming the request on the client end, it is sent to the server and passed to the Mail Manager. Information such as the sender, receiver and date are stored for each incoming message for later use. The message file is manipulated using the File Manager Upload function to upload the file from the client into the Mail Depository.