CS 4323 Design and Implementation of Operating Systems I

Assignment 02: Full Marks 100
(Due Date and End date: 03/28/2021, 11:59 PM CST)

This is the mini project assignment that must be done in the group as specified using C programming language. If you do not know the group members, then please refer the module **MiniGroupProjectFormation** under **Home** page in Canvas.

This project will familiarize you with Unix socket programming for a simple client-server interaction and inter-process communication (IPC) using pipe.

You are required to write a multithreaded server capable of receiving messages (or request) from the client, process the request and send the result back to the client. The server does not need to print anything on the screen. The user is going to interact with the processes at the client side.

Input file:

There is an input file with name input.xlsx (which is an excel file). It consists of three sheets: **ID_Name**, **SatisfactionLevel** and **Salaries**.

- The sheet **ID_Name** consists of the EmployeID (given by ID) and the corresponding name.
- The sheet **SatisfactionLevel** consists of several information which are related to determine whether the employee is satisfied with the job or not.
    - Column *ID* is the employee ID
    - Column *satisfaction_level* is the index between 0 and 1, to indicate how satisfied the employee is. Value 0 indicates least satisfied while value 1 indicates very satisfied.
    - Column *number_project* indicates the number of projects the employee has undertaken during the employment time.
    - Column *average_monthly_hours* indicates the number of hours employee works on average in a month.
    - Column *time_spend_company_in_yrs* indicates the employment period.
    - Column *work_accident* indicates number of accidents the employee had during the employment period.
    - Column *promotion_last_5years* indicates how many promotions the employee had in the past 5 years.
- The sheet **Salaries** consists of several information which are related to salary:
    - Column *ID* is the employee ID
    - Column *JobTitle* is the the employee Job title
    - Column *BasePay* is the base salary of the employee
    - Column *OvertimePay* is the salary received by the employee working overtime
    - Column *Benefit* is the additional payment received by the employee.

- Column *Status* indicates whether the employee is full time or part time.

You are allowed to make three separate txt files: one for each sheet. However, you are not allowed to change any content in the file.

Client Side:

At the client's side, there are two processes running concurrently:

- the Manager which is the only process that will interacting with the user.
- the Assistant which will be responsible to interact with the server.

The user enters *EmployeeName*, *JobTitle* and *Status* as an input to the Manager. The Manager takes this input and sends the employee name as a query to the Assistant using pipe. The Manager then immediately becomes ready to take another input from the user, while Assistant works in the background on the given query. The Assistant first checks its local file to see if there is any information for the given query or not. If the file contains the result for the given query then the Assistant will give the result from its local file. If the file does not contain the data for the given query, only then the Assistant will make connection to the server and passes the query. Till the Assistant does not get result back from the server, it will not send any additional query to the server. So, there are few points to note:

- the Manager is continuously interacting with the users and hence can receive many queries. It will send queries one by one to the Assistant. Assistant can receive those queries in the same order as it is sent by the Manager.
- the Assistant initially maintains an empty file (let us call it a history file). This file is used to store the employee record, that has been received from the server. For example, for the first time, the history file is empty. If the query is for the employee ALSON LEE, then Assistant will first go and look at this history file. Since the history file does not contain the requested result, the Assistant forwards the query to the server and waits for the result. When the Assistant receives the valid result, it stores that result in the history file. Next time, when the Assistant receives the query for ALSON LEE, it does not need to go to the server. It can access its local history file and gives the result to the user.
- the Assistant sends only one request to the server and then waits for the result from the server, before it sends another request.

The purpose of history file is to provide quick result to the user. However, the history file can only maintain the record for only 10 employees at any time. Once there are 10 employees in the history file, the 11^{th} employee information would replace the record, which appeared first in the history file. This means:

- 11^{th} employee information will replace 1^{st} employee record,
- 12^{th} employee information will replace 2^{nd} employee record,
- And so on..

The history file will contains all the information regarding the query. For example if query is for ALSON LEE, then its contains following information about ALSON LEE in a single line:

- Id
- EmployeeName
- JobTitle
- BasePay
- OvertimePay
- Benefit
- Status
- satisfaction_level
- number_project
- average_monthly_hours
- time_spend_company_in_yrs
- Work_accident
- promotion_last_5years

This implies that the server needs to send all of the above information to the Assistant as a result for any given request.

Once the Assistant receives the request from the server, it will output the result to the new terminal. This implies that:

- the Manager will only accept the inputs from the user and forwards them to the Assistant
- the Assistant will output the results either directly from the history file, if the result is available there, or first gets the result from the server and then displays it.
- The Assistant will display the result in the new terminal (different from the one that is used to take input from the user). This new terminal is used only to display the results.

Server Side:

The server will use the input file. When the server process receives a request from the Assistant, the main thread will first find the corresponding *ID* for the given *EmployeeName*. Once the *ID* is found, two threads are created to search the two sheets:

- One thread (lets called its Satisfaction) will search in the sheet **SatisfactionLevel**
- Another thread (lets called its Salaries) will search in the sheet **Salaries**

Note: You need to consider there can be two employees with the same name. That is why, the Manager requests for *EmployeeName*, *JobTitle* and *Status* as an input in the first place. If there are two employees with exactly the same *EmployeeName*, *JobTitle* and *Status*, then you can choose any employee.

The results from the 2 threads need to be passed to the main thread and it is the main thread that will send the result back to the Assistant.

Points distribution for this project will be as follows:

- Message passing between the Manager and the Assistant processes using pipe. **[20 points]**

- Display of result in different terminal by Assistant process. **[20 points]**

- Implementation of History file by Assistant process. **[10 points]**

- Implementation of message passing between client-server processes **[10 points]**
    - Client-side processes and the server process should not necessarily be running in the same machine

- Implementation of multi-threaded server for searching the record in the sheets **SatisfactionLevel** and **Salaries**. **[15 points]**

- Information passing between the main thread and Satisfaction and between main thread and Salaries. **[15 points]**

- Progress report and group coordination. **[10 points]**

Grading Criteria:

This is a group project. So, grading will focus on students' ability to work in the group and successfully complete the work. Each member in the group should coordinate as a team rather than individual. And that's the key to score maximum points.

Points to remember:

- Failure to complete the project as a group will deduct 20 point, even though every individual has completed their part. As this is a group project, you need to learn to work in the group and meet the team's objective, and not individual objective.
- Work must be distributed uniformly among all group member and should be clearly specified in the report.
    - Be sure to submit a progress report that summarizes your design and all the test cases you have performed.
    - The progress report needs to include the code and the output.
        - Each student will write the code in their own file. So, if a group has 4 members, then there should be 5 .c files and appropriate number of .h file(s). Each .c file belong to one group member.
    - Include each student's contribution clearly. Failure to be specific on this part will make us difficult to give fair grades. So, it is your responsibility to clearly indicate what specific task each member is undertaking in the project.

- It is not necessary that all group members will get equal points. It depends on what responsibility each student has taken, whether or not the student has delivered the task.
- It is the group's responsibility to alert instructor with any issue that is happening in the group.
    - Students are supposed to use the group created in the Canvas for all correspondence. That will serve as the proof, in case there is any dispute among the group members.
    - Time is very important. Group members are expected to respond quickly.
    - You need to have sufficient days to combine individual work. You are going to make one submission as a group and not individually.

Submission Guidelines:

- Progress report need to be submitted by the progress due date/end date (03/17/2021, 1:59 PM CDT). The due date and the end date are same for the progress report. During the progress report, each group need to submit:
    - Please refer to the progress report file for further instruction
- Final project submission instruction:
    - Include a single readMe.txt file that shows how to run your program.
    - Each student's work needs to be in a separate .c file.
        - E.g.: If the group has 4 members, then there should be 5 .c files: each .c file representing one of the group members and the fifth .c file will be the driver file (i.e. file containing main function).
        - You can have .h file.
    - Each .c file should include the header information, which should include:
        - Group Number
        - Group member name
        - Email
    - There should be sufficient comments in the program.
        - Each function should be clearly described along with the input arguments and return values.
    - All the codes should be submitted in both .pdf format (copy paste in textual form, no screenshot) as well as in .c files (including any .h file).
- Final project will be evaluated together with the progress report. So, they should have a correlation. Commitments on the progress report should be reflected on the final submissions.