Distributed Project

Assignment due dates:

Project description: 9 pm on Friday, April 24th. Use D2L. Submit a **`description.pdf`** or **`description.txt`** file.

Project submission: Turnin the code and the report by 9 p.m. on Wednesday, May 6th.

Project demonstration: These will be scheduled for Friday to Tuesday, May 8th to 12th.

Description:

You are to design and implement a concurrent program that makes creative use of several processes running on a distributed set of computers. You may use C with MPI, C with sockets, Java with sockets, or Java with RMI, or combinations of these. You may substitute other flavors of C; i.e., C++, Objective-C.

The choice of the program is purposely unspecified; pick something that you think would be interesting and educational.

Exercise 7.26, pages 359-360, describes several possible programming projects (see the Appendix to this assignment). Other projects are possible. For example, you could take a parallel solution to the first project (or some similar parallel application) and add an animation window to it. The animation code would run on one computer, the parallel computation on a set of different computers. Games are another source of ideas; card games have been popular choices in the past. Animations that require multiple machines and/or displays are another source. Consider, for example, a game or animation where objects move across multiple displays each attached to a different machine. Games that allow participation from more than one machine are another example. Another graphical-computation problem would be a distributed fractal solution that spreads the work across multiple machines.

Machines available for this project include: lectura and the machines in the 228 and 930 labs. You can use any combination of these machines as appropriate for your project.

Unlike the previous project, there is not a requirement for performance measurements. If you want to do a project that is performance-based, talk with me about the machines needed and we will see about setting up reservation slot(s) for timing runs.

The emphasis here is on the distributed computing aspects of the project. There should be multiple machines involved (> 2 machines). If you are using a central server idea, the server would be in addition to the > 2 machines requirement.

Project teams:

You may work solo or in teams of two. Two person teams are expected to undertake a more ambitious, or more polished, project. Team membership can be the same as for the parallel project or different.

Project description: Wednesday, April 22nd

Give me a brief (one to three paragraphs) description of what you propose to do, and the members (solo or group) of the team.

I will give you feedback by the 26th. I can respond earlier if you provide an earlier description. You are welcome and encouraged to discuss possible topics with me (email, in person, phone) before the 22nd.

Final report and demonstration: no later than Tuesday, May 12th

You will demonstrate your project for me. You will need to prepare the demonstration. Plan a presentation that will last about 15 minutes. This will be followed by my questions. The total time for demonstration and questions will be no longer than 25 minutes. If doing a two-person project, both members of the group must be present. Sign-ups sheet will be available for times on Friday, Monday, and Tuesday, May 8th, 11th, and 12th. You can also make an appointment for a demonstration time prior to the 8th.

Demonstrations can be done in my office, in GS-228, or in GS-930. The demonstration can also involve personal laptops.

At the time of your demonstration, provide:

- a written description of your project — not more than 5 pages.
- Include in your report a diagram of your program that shows how it is distributed across the machines you are using and the communication among the machines/processes.

Turnin:

Turn in your programs. Include a **`Makefile`** that supports the command **`make all`** that will create the executable or class files for your program. You do not need to turn in the actual output from any of your tests. Instead, turnin a **`README.txt`** file that gives examples of commands that you used in your tests. The idea is to tell us what commands so we can duplicate your results. We will not exhaustively test your program. We will selectively run tests to verify that your program works as you describe.

Note: We are asking for submission of programs via D2L only.

- Submit all source files. Include a **`Makefile`** with the target **`all`**. The command “make all” should produce a compiled C executable or all needed Java class files. For the report use **`report.txt`** or **`report.pdf`**.
- Logon to D2L. Select CSc 422. Click on the “Add a File” button. A pop-up window will appear. Click on the “Choose File” button. Use the file browser that will appear to select your file(s). Click on the “Upload” button in the lower-right corner of the pop-up window.
- You are now back at “Submit Files ”. Click on “Upload” in the lower-right of this window. You should now be at the “File Upload Results” page, and should see the message **`File Submission Successful`**.

D2L will send you a confirmation email after each item is placed in the Dropbox. This email is sent to your UA email (which is <yourUANetId>@email.arizona.edu). Retain these emails at least until you have received a grade for the assignment; it is your confirmation that you did submit the program(s).

You may submit the various items more than once. We will grade the last one that you put in the Dropbox.

Appendix:

Exercise 7.26, pages 359-360:

Design, implement, and document a parallel or distributed program that makes creative use of several processes. Use a programming language or subroutine library that is available on your local installation.

The following is a list of possible projects. Either pick one of these or design your own project having a comparable level of difficulty. Provide your instructor with a brief description of your project before you begin, and demonstrate your program when the project is completed.

a.) Take some problem that could be parallelized or distributed in different ways - e.g., sorting, prime number generations, traveling sales rep, matrix multiplication. Develop different algorithms to solve the problem, and perform a series of experiments to determine which approach works best under different assumptions. Ideally, perform experiments on both a shared-memory multiprocessor and a distributed-memory machine.

b.) Construct an automated teller system having several customer "terminals" and several banks. Each ATM is connected to one bank, and the banks are connected to each other. Each account is stored at one bank, but a customer should be able to use an ATM to access his or her account. Your program should implement several kinds of user transactions, such as deposits, withdrawals, and balance inquiries. Ideally, construct a user interface that looks like a teller machine.

c.) Construct an airline reservation system that allows users to ask about flights and to reserve and cancel seats. Support multiple reservation agents. This project is similar to implementing an automated teller system.

d.) Develop a program that plays some game and employs several displays, either to show the results for different players or to enable several players to play against each other and perhaps "the machine." If you choose this project, keep the game relatively simple so that you do not get bogged down just getting the game itself programmed. Fro example, implement a simple card game such as (basic) blackjack or concentration, or a simple video game.

e.) Implement a "talk" or "chat" command that allows users on different terminals to converse with each other. Support both private conversations and conference calls. Allow a person to join a conversation that is already in progress.

f.) Develop a discrete-event simulation of some physical system. For example, simulate traffic moving through an intersection, planes arriving and departing at an airport, or customers riding elevators in a building. Implement the various entities in the simulation as processes that generate events and react to them. You may want to have the processes interact directly with each other as events occur, or you may want to use a scheduler process to keep track of an event list and update a simulation clock. Animate your program so that users can see the simulation execute and possibly interact with it.