Lista 6 - Capítulo 8 - RPC e Randevous

8.1 Modify the time server module in Figure 8.1 so that the clock process does not get awakened on every tick of the clock. Instead, the clock process should set the hardware timer to go off at the next interesting event. Assume that the time of day is maintained in milliseconds and that the timer can be set to any number of milliseconds. Also assume that processes can read how much time is left before the hardware timer will go off. Finally, assume that the timer can be reset at any time.

8.6 Consider a self-scheduling disk driver process as in Figure 7.9. Suppose the process exports one operation: **`request(cylinder, ...)`**. Show how to use rendezvous and an **`in`** statement to implement each of the following disk-scheduling algorithms: shortest seek time, circular scan, and elevator. (*Hint:* Use scheduling expressions.)

8.8 Consider the following specification for a program to find the minimum of a set of integers. Given is an array of processes **`Min[1:n]`**. Initially, each process has one integer value. The processes repeatedly interact, with each one trying to give another the minimum of the set of values it has seen. If a process gives away its minimum value, it terminates. Eventually, one process will be left, and it will know the minimum of the original set.

(a) Develop a program to solve this problem using only the RPC primitives defined in Section 8.1.

(b) Develop a program to solve this problem using only the rendezvous primitives defined in Section 8.2.

(c) Develop a program to solve this problem using the multiple primitives defined in Section 8.3. Your program should be as simple as you can make it.

8.14 *The Savings Account Problem.* A savings account is shared by several people. Each person may deposit or withdraw funds from the account. The current balance in the account is the sum of all deposits to date minus the sum of all withdrawals to date. The balance must never become negative.

Using the multiple primitives notation, develop a server to solve this problem and show the client interface to the server. The server exports two operations: **`deposit(amount)`** and **`withdraw(amount)`**. Assume that **`amount`** is positive and that **`withdraw`** must delay until there are sufficient funds.

8.16 Suppose a computer center has two printers, `A` and `B`, that are similar but not identical. Three kinds of client processes use the printers: those that must use `A`, those that must use `B`, and those that can use either `A` or `B`.

Using the multiple primitives notation, develop code that each kind of client executes to request and release a printer, and develop a server process to allocate the printers. Your solution should be fair, assuming a client using a printer eventually releases it.