First Readers Writers Problem

Introduction

This algorithm uses a monitor to solve the first readers-writers problem.

Algorithm

```
 1   readerwriter: monitor
 2   var     readcount: integer;
 3                   writing: boolean;
 4                   oktoread, oktowrite: condition;
 5   procedure entry     beginread;
 6   begin
 7           readcount := readcount + 1;
 8           if writing then
 9                   oktoread.wait ;
10   end ;
11   procedure   entry   endread;
12   begin
13           readcount := readcount - 1;
14           if readcount = 0 then
15                   oktowrite.signal  ;
16           end ;
17   procedure   entry   beginwrite;
18   begin
19           if readcount > 0 or writing then
20                   oktowrite.wait ;
21           writing := true;
22   end ;
23   procedure   entry   endwrite;
24   var     i: integer;
25   begin
26           writing := false;
27           if readcount > 0 then
28                   for  i := 1 to readcount
29                           oktoread.signal  ;
30           else
31           oktowrite.signal;
32   end ;
33   begin
34           readcount := 0; writing := false;
35   end .
```

Comments

lines 1-4 Here, `readcount` contains the number of processes reading the file, and `writing` is true when a writer is writing to the file. `Oktoread` and `oktowrite` correspond to the logical conditions of being able to access the file for reading and writing, respectively.

lines 7-9 In this routine, the reader announces that it is ready to read (by adding 1 to `readcount`). If a writer is accessing the file, it blocks on the condition variable `oktoread`; when done, the writer will signal on that condition variable, and the reader can proceed.

lines 13-15 In this routine, the reader announces that it is done (by subtracting 1 from `readcount`). If no more readers are reading, it indicates a writer may go ahead by signalling on the condition variable

oktowrite.

lines 19-21 In this routine, the writer first sees if any readers or writers are accessing the file; if so, it waits until they are done. Then it indicates that it is writing to the file by setting the boolean *writing* to `true`.

lines 26-31 Here, the writer first announces it is done by setting `writing` to `false`. Since readers have priority, it then checks to see if any readers are waiting; if so, it signals all of them (as many readers can access the file simultaneously). If not, it signals any writers waiting.

line 34 This initializes the variables.