```
module FileServer[myid = 1 to n]
  type mode = (READ, WRITE);
  op open(mode), close(),        # client operations
     read(result  result types), write(value types);
  op startwrite(), endwrite(),   # server operations
     remote_write(value types);
body
  op startread(), endread();    # local operations
  mode use;  declarations for file buffers;

  proc open(m) {
    if (m == READ) {
      call startread();     # get local read lock
      use = READ;
    } else {     # mode assumed to be WRITE
      # get write locks for all copies
      for [i = 1 to n]
        call FileServer[i].startwrite();
      use = WRITE;
    }
  }

  proc close() {
    if (use == READ)    # release local read lock
      send endread();
    else    # use == WRITE, so release all write locks
      for [i = 1 to n]
        send FileServer.endwrite()
  }

  proc read(results) {
    read from local copy of file and return results;
  }

  proc write(values) {
    if (use == READ)
      return with error:  file was not opened for writing;
    write values into local copy of file;
    # concurrently update all remote copies
    co [i = 1 to n st i != myid]
      call FileServer[i].remote_write(values);
  }

  proc remote_write(values) { # called by other servers
    write values into local copy of file;
  }
```

```
  process Lock {
    int nr = 0, nw = 0;
    while (true) {
      ## RW: (nr == 0 ∨ nw == 0) ∧ nw <= 1
      in startread() and nw == 0 -> nr = nr+1;
      [] endread() -> nr = nr-1;
      [] startwrite() and nr == 0 and nw == 0 ->
             nw = nw+1;
      [] endwrite() -> nw = nw-1;
      ni
    }
  }
end FileServer
```

Figure 8.15 Replicated files using one lock per copy.