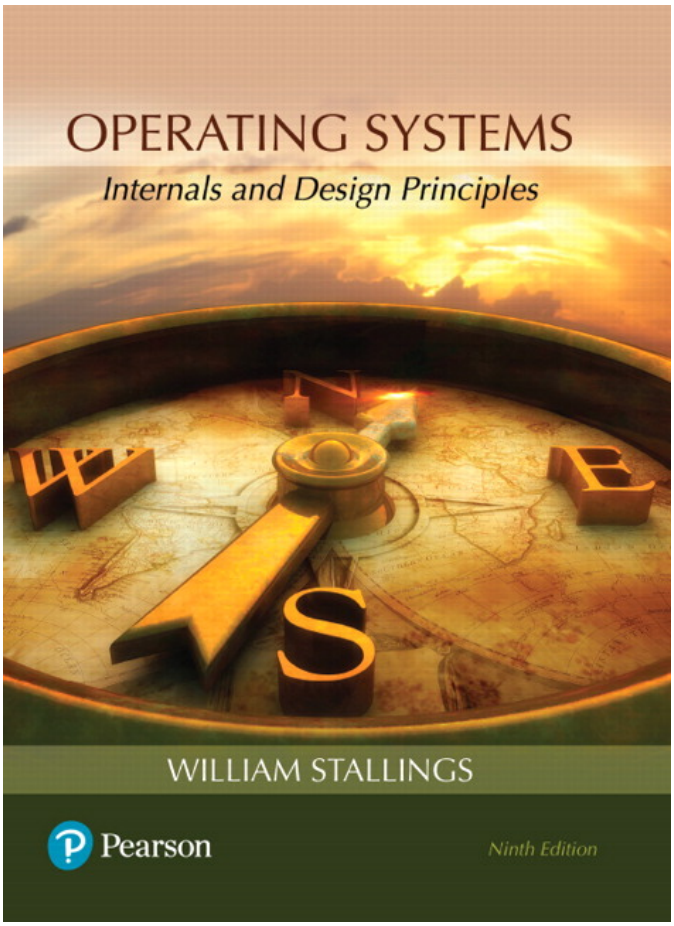


William Stallings

Operating Systems

Internals and Design Principles

Ninth Edition
2017

Readers/Writers Problem, Ver. 3

rdr_wrt_msg_v4.pml (1/7)

```
$ cat -n rdr_wrt_msg_v4.pml | expand
     1  #define NRDRS   5
     2  #define NWRTS   2
     3  #define MAXRDRQ 20
     4  #define MAXWRRQ 20
     5
     6  chan readrequest  = [MAXRDRQ] of { byte }
     7  chan writerequest = [MAXWRRQ] of { byte }
     8  chan finished     = [MAXRDRQ+MAXWRRQ] of { byte }
     9  chan mbox[NRDRS+NWRTS+1] = [MAXRDRQ+MAXWRRQ] of { bool }
    10
    11  int count = 100
    12  mtype = { reader, writer }
    13  byte nr = 0, nw = 0
    14
...
```

Seguimos "parchando" los errores ...

rdr_wrt_msg_v4.pml (2/7)

```
...
    15  proctype ReaderWriter(byte i; mtype who) {
    16      chan ch
    17      if
    18      :: who == reader -> ch = readrequest
    19      :: else -> ch = writerequest
    20      fi
    21
    22      ch ! i
    23      atomic {
    24          mbox[i] ? _
    25          printf("%e %d\n",who,i)
    26      }
...
```


rdr_wrt_msg_v4.pml (3/7)

```
...
    27      if
    28      :: who == reader -> nr++
    29      :: else -> nw++
    30      fi
    31      assert(nw < 2)
    32      assert((nw > 0 && nr == 0) || (nw == 0 && nr > 0))
    33      atomic {
    34          if
    35          :: who == reader -> nr--
    36          :: else -> nw--
    37          fi
    38          finished ! i
    39      }
    40  }
    41
...
```

rdr_wrt_msg_v4.pml (4/7)

```
...
    42  proctype Controller() {
    43      byte p
    44
    45      do
    46      ::  count > 0 ->
    47  end:    if
    48          ::  nempty(finished) ->
    49                  atomic {
    50                      finished ? p
    51                      printf("finished %d\n",p)
    52                  }
    53                  count++
    54          ::  empty(finished) && nempty(writerequest) ->
    55                  atomic {
    56                      writerequest ? p
    57                      printf("request from Writer %d\n",p)
    58                  }
    59                  count = count - 100
...
```


rdr_wrt_msg_v4.pml (5/7)

```
...
    60          ::  empty(finished) && empty(writerequest) && nempty(readrequest) ->
    61                  atomic {
    62                      readrequest ? p
    63                      printf("request from Reader %d\n",p)
    64                  }
    65                  count--
    66                  atomic {
    67                      mbox[p] ! true
    68                      printf("OK to Reader %d\n",p)
    69                  }
    70          fi
    71      ::  count == 0 ->
    72              atomic {
    73                  mbox[p] ! true
    74                  printf("OK to Writer %d\n",p)
    75              }
    76              atomic {
    77                  finished ? p
    78                  printf("finished Writer %d\n",p)
    79              }
    80              count = 100
```

rdr_wrt_msg_v4.pml (6/7)

```
...
    81      ::  count < 0 ->
    82              atomic {
    83                  finished ? p
    84                  printf("finished Writer %d\n",p)
    85              }
    86              count++
    87      od
    88  }
    89
...
```


rdr_wrt_msg_v4.pml (7/7)

```
...
    90  init {
    91      byte i
    92
    93      atomic {
    94          for (i : 1 .. NRDRS+NWRTS) {  /* R1,R2,W3,R4,W5,R6,R7 */
    95              if
    96              ::  i == 3 || i == 5 ->
    97                      run ReaderWriter(i,writer)
    98              ::  else ->
    99                      run ReaderWriter(i,reader)
   100              fi
   101          }
   102          run Controller()
   103      }
   104  }
```

Verification: 1 error

```
$ spin -run rdr_wrt_msg_v4.pml | expand
pan:1: invalid end state (at depth 137)
pan: wrote rdr_wrt_msg_v4.pml.trail

(Spin Version 6.4.6 -- 2 December 2016)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +

State-vector 572 byte, depth reached 166, errors: 1
...
```


Invalid End State Error trail (1/3)

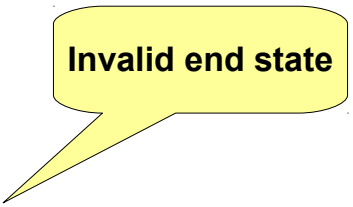
```
$ spin -t rdr_wrt_msg_v4.pml | expand
                                        request from Reader 7
                                        OK to Reader 7
                                   reader 7
                                        finished 7
                                        request from Reader 6
                                        OK to Reader 6
                                reader 6
                                        finished 6
                                        request from Writer 5
                                        OK to Writer 5
                            writer 5
                                        finished Writer 5
                                        request from Reader 4
                                        OK to Reader 4
                        reader 4
                                        finished 4
                                        request from Reader 2
                                        OK to Reader 2
                                        request from Writer 3
...
```

Invalid End State Error trail (2/3)

```
...
              reader 2
                                        finished Writer 2
                                        OK to Writer 2
spin: trail ends after 138 steps
#processes: 9
                queue 1 (readrequest): [1]
                queue 3 (writerequest):
                queue 2 (finished):
                queue 4 (mbox[0]):
                queue 5 (mbox[1]):
                queue 6 (mbox[2]): [1]
                queue 7 (mbox[3]):
                queue 8 (mbox[4]):
                queue 9 (mbox[5]):
                queue 10 (mbox[6]):
                queue 11 (mbox[7]):
                count = 100
                nr = 0
                nw = 0
...
```


Invalid End State Error trail (3/3)

```
...
```




```
138:    proc  8 (Controller:1) rdr_wrt_msg_v4.pml:76 (state 28)
138:    proc  7 (ReaderWriter:1) rdr_wrt_msg_v4.pml:40 (state 27) <valid end state>
138:    proc  6 (ReaderWriter:1) rdr_wrt_msg_v4.pml:40 (state 27) <valid end state>
138:    proc  5 (ReaderWriter:1) rdr_wrt_msg_v4.pml:40 (state 27) <valid end state>
138:    proc  4 (ReaderWriter:1) rdr_wrt_msg_v4.pml:40 (state 27) <valid end state>
138:    proc  3 (ReaderWriter:1) rdr_wrt_msg_v4.pml:23 (state 10)
138:    proc  2 (ReaderWriter:1) rdr_wrt_msg_v4.pml:40 (state 27) <valid end state>
138:    proc  1 (ReaderWriter:1) rdr_wrt_msg_v4.pml:23 (state 10)
138:    proc  0 (:init::1) rdr_wrt_msg_v2.pml:104 (state 17) <valid end state>
9 processes created
```

Original Controller's code (1/6)

```
 1 void controller()
 2 {
 3     while (true)
 4     {
 5         if (count > 0) {
 6             if (!empty (finished)) {
 7                 receive (finished,msg);
 8                 count++;
 9             }
10             else if (!empty (writerequest)) {
11                       receive (writerequest,msg);
12                       writer_id = msg.id;
13                       count = count - 100;
14                  }
15                  else if (!empty (readrequest)) {
16                           receive (readrequest,msg);
17                           count--;
18                           send (mbox[msg.id],"OK to proceed");
19                       }
20         }
21         if (count == 0) {
22             send (mbox[writer_id],"OK to proceed");
23             receive (finished,msg);
24             count = 100;
25         }
26         while (count < 0) {
27             receive (finished,msg)
28             count++;
29         }
30     }
31 }
```