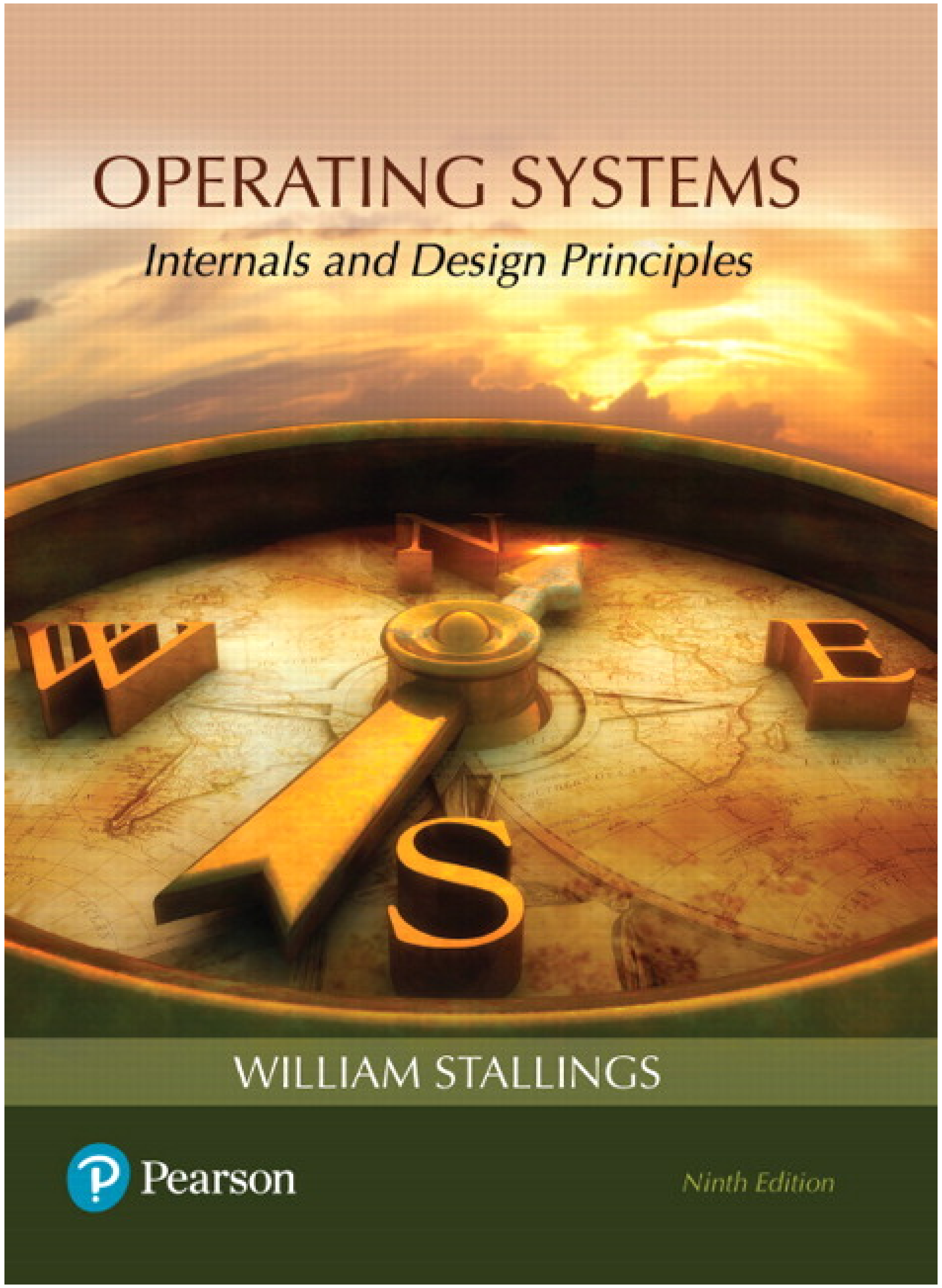

William Stallings

Operating Systems

Internals and Design Principles

Ninth Edition
2017

Readers/Writers Problem, Ver. 4

Original Controller's code

```
void controller()
{
    while (true)
    {
        if (count > 0) {
            if (!empty (finished)) {
                receive (finished,msg);
                count++;
            }
            else if (!empty (writerequest)) {
                     receive (writerequest,msg);
                     writer_id = msg.id;
                     count = count - 100;
                 }
                 else if (!empty (readrequest)) {
                          receive (readrequest,msg);
                          count--;
                          send (mbox[msg.id],"OK to proceed");
                      }
        }
        if (count == 0) {
            send (mbox[writer_id],"OK to proceed");
            receive (finished,msg);
            count = 100;
        }
        while (count < 0) {
            receive (finished,msg)
            count++;
        }
    }
}
```

`rdr_wrt_msg_v5.pml` (1/8)

```
$ cat -n rdr_wrt_msg_v5.pml | expand

     1  #define NRDRS   5
     2  #define NWRTS   2
     3  #define TOTALT  20
     4
     5  chan readrequest  = [NRDRS] of { byte }
     6  chan writerequest = [NWRTS] of { byte }
     7  chan finished     = [NRDRS+NWRTS] of { byte }
     8  chan mbox[NRDRS+NWRTS] = [1] of { bool }
     9
    10  mtype = { reader, writer }
    11  byte start[NRDRS+NWRTS]
    12  byte nr = 0, nw = 0
    13  byte t = 0
    14
...
```

`rdr_wrt_msg_v5.pml` (2/8)

```
...
proctype ReaderWriter(byte i; mtype who) {
    chan ch

    atomic {
        if
        :: who == reader -> ch = readrequest
        :: else -> ch = writerequest
        fi
        (start[i] <= t)
        ch ! i
        mbox[i] ? _
        printf("t=%d: %e %d\n",t,who,i)
        if
        :: who == reader -> nr++
        :: else -> nw++
        fi
        t++
    }
...
```

rdr_wrt_msg_v5.pml (3/8)

```
...
    assert(nw < 2)
    assert((nw > 0 && nr == 0) || (nw == 0 && nr > 0))
    atomic {
        if
        :: who == reader ->
            (t >= start[i]+2)
            nr--
        :: else ->
            (t >= start[i]+5)
            nw--
        fi
        finished ! i
    }
}

...
```

rdr_wrt_msg_v5.pml (4/8)

```
...
proctype Controller() {
    byte r, w, rdrcount=0

end:
    do
    ::  nempty(finished) || nempty(writerequest) || nempty(readrequest)
        if
        ::  nempty(finished) ->
                atomic {
                    finished ? r
                    printf("t=%d: finished Reader %d\n",t,r)
                    rdrcount--
                }
        ::  empty(finished) && nempty(writerequest) ->
                atomic {
                    writerequest ? w
                    printf("t=%d: request from Writer %d\n",t,w)
                }
...
```

`rdr_wrt_msg_v5.pml` (5/8)

```
...
        do
        ::  rdrcount == 0 -> break
        ::  else ->
                atomic {
                    finished ? r
                    printf("t=%d: finished Reader %d\n",t,r)
                    rdrcount--
                }
        od
        atomic {
            mbox[w] ! true
            printf("t=%d: OK to Writer %d\n",t,w)
        }
        atomic {
            finished ? w
            printf("t=%d: finished Writer %d\n",t,w)
        }
...
```

rdr_wrt_msg_v5.pml (6/8)

```
...
        ::  empty(finished) && empty(writerequest) &&
            nempty(readrequest) ->
                atomic {
                    readrequest ? r
                    printf("t=%d: request from Reader %d\n",t,r)
                    rdrcount++
                }
                atomic {
                    mbox[r] ! true
                    printf("t=%d: OK to Reader %d\n",t,r)
                }
        fi
    od
}

...
```

rdr_wrt_msg_v5.pml (7/8)

```
...
proctype Idle() {
    do
    ::  atomic {
            timeout ->
                if
                ::  t >= TOTALT || _nr_pr == 3 -> break
                ::  else ->
                        printf("t: %d -> %d\n",t,t+1)
                        t++
                fi
        }
    od
}

...
```

rdr_wrt_msg_v5.pml (8/8)

```
...
init {
    byte i

    atomic {
        run Controller()
        for (i : 0 .. NRDRS+NWRTS-1) {  /* R0,R1,W2,R3,R4,W5,R6 */
            if
            ::  i == 2 || i == 5 ->
                    run ReaderWriter(i,writer)
            ::  else ->
                    run ReaderWriter(i,reader)
            fi
            start[i] = i
        }
    }
    run Idle()
    (_nr_pr == 2)
}
```

Random simulation (1/4)

```
$ spin rdr_wrt_msg_v5.pml | expand
          t=0: request from Reader 0
          t=0: OK to Reader 0
              t=0: reader 0
          t=1: request from Reader 1
          t=1: OK to Reader 1
                  t=1: reader 1
          t=2: finished Reader 0
          t=2: request from Writer 2
      timeout
                                            t: 2 -> 3

          t=3: finished Reader 1
          t=3: OK to Writer 2
                      t=3: writer 2
      timeout
                                            t: 4 -> 5

      timeout
                                            t: 5 -> 6

      timeout
                                            t: 6 -> 7

...
```

Random simulation (2/4)

```
...
          t=7: finished Writer 2
          t=7: request from Writer 5
          t=7: OK to Writer 5
                                  t=7: writer 5
      timeout
                                          t: 8 -> 9
      timeout
                                          t: 9 -> 10
          t=10: finished Writer 5
          t=10: request from Reader 3
          t=10: OK to Reader 3
                          t=10: reader 3
          t=11: finished Reader 3
          t=11: request from Reader 4
          t=11: OK to Reader 4
                              t=11: reader 4
          t=12: finished Reader 4
          t=12: request from Reader 6
          t=12: OK to Reader 6
                                      t=12: reader 6
          t=13: finished Reader 6
```

Random simulation (3/4)

```
...
      timeout
                                            t: 13 -> 14
      timeout
                                            t: 14 -> 15
      timeout
                                            t: 15 -> 16
      timeout
                                            t: 16 -> 17
      timeout
                                            t: 17 -> 18
      timeout
                                            t: 18 -> 19
      timeout
                                            t: 19 -> 20
      timeout
      timeout
#processes: 2
                queue 1 (readrequest):
                queue 2 (writerequest):
                queue 3 (finished):
...
```

Random simulation (4/4)

```
...
                        queue 4 (mbox[0]):
                        queue 5 (mbox[1]):
                        queue 6 (mbox[2]):
                        queue 7 (mbox[3]):
                        queue 8 (mbox[4]):
                        queue 9 (mbox[5]):
                        queue 10 (mbox[6]):
                        start[0] = 0
                        start[1] = 1
                        start[2] = 2
                        start[3] = 3
                        start[4] = 4
                        start[5] = 5
                        start[6] = 6
                        nr = 0
                        nw = 0
                        t = 20
305:    proc  1 (Controller:1) rdr_wrt_msg_v5.pml:52 (state 37) <valid end state>
305:    proc  0 (:init::1) rdr_wrt_msg_v5.pml:128 (state 20) <valid end state>
10 processes created
```

Verification: 0 error

```
$ spin -run rdr_wrt_msg_v5.pml | expand

(Spin Version 6.4.6 -- 2 December 2016)
        + Partial Order Reduction

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

State-vector 196 byte, depth reached 205, errors: 0
     5333 states, stored
    13968 states, matched
    19301 transitions (= stored+matched)
    18461 atomic steps
hash conflicts:         0 (resolved)

...
```