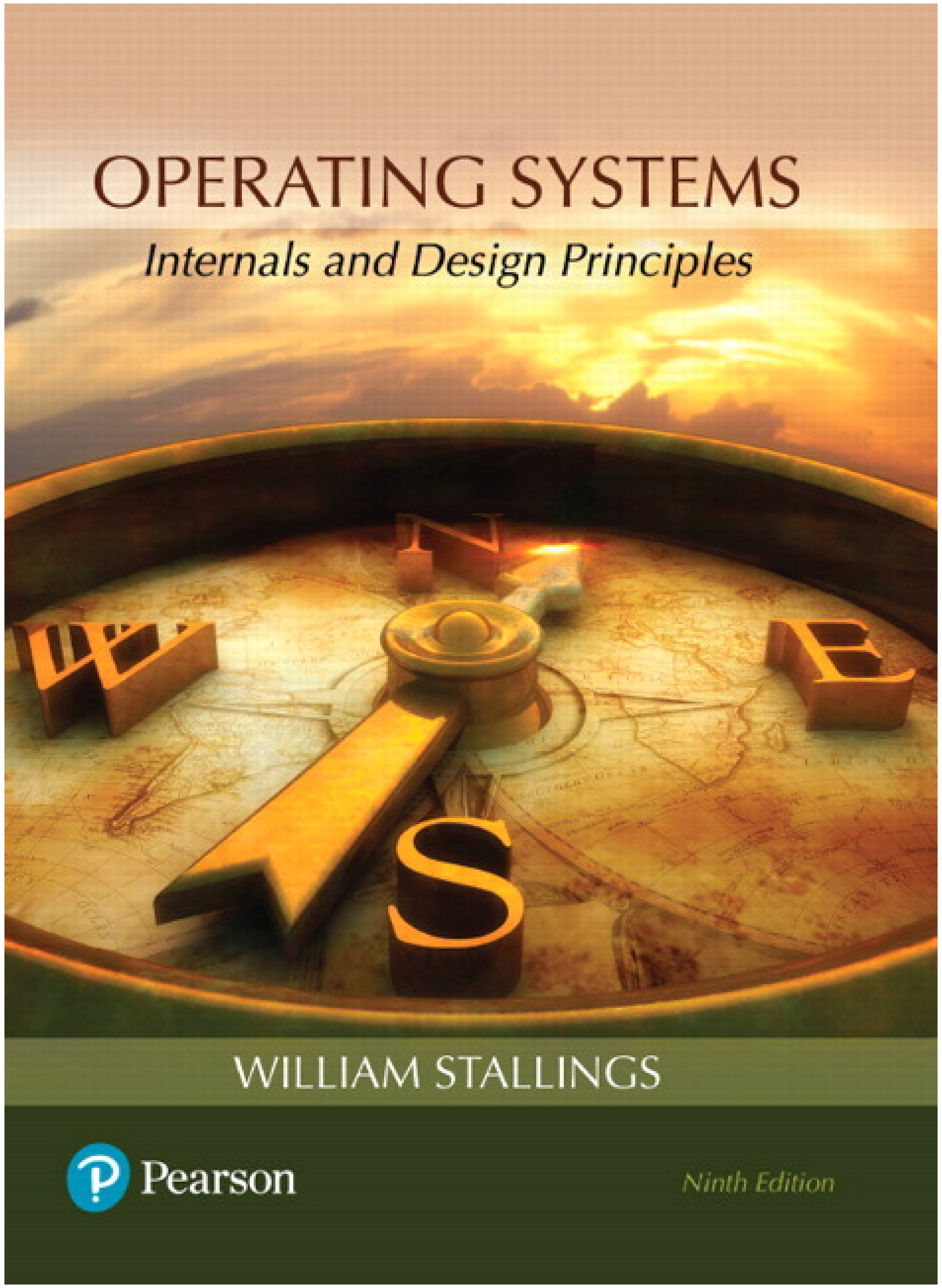

William Stallings

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)

```
...
proctype ReaderWriter(byte i; mtype who) {
    chan ch
    if
    :: who == reader -> ch = readrequest
    :: else -> ch = writerequest
    fi

    ch ! i
    atomic {
        mbox[i] ? _
        printf("%e %d\n",who,i)
    }
...
```

rdr_wrt_msg_v4.pml (3/7)

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

...
```

rdr_wrt_msg_v4.pml (4/7)

```
...
proctype Controller() {
    byte p

    do
    ::  count > 0 ->
end:    if
        ::  nempty(finished) ->
                atomic {
                    finished ? p
                    printf("finished %d\n",p)
                }
                count++
        ::  empty(finished) && nempty(writerequest) ->
                atomic {
                    writerequest ? p
                    printf("request from Writer %d\n",p)
                }
                count = count - 100
...
```

rdr_wrt_msg_v4.pml (5/7)

```
...
            :: empty(finished) && empty(writerequest) && nempty(readrequest) ->
                   atomic {
                       readrequest ? p
                       printf("request from Reader %d\n",p)
                   }
                   count--
                   atomic {
                       mbox[p] ! true
                       printf("OK to Reader %d\n",p)
                   }
            fi
        :: count == 0 ->
               atomic {
                   mbox[p] ! true
                   printf("OK to Writer %d\n",p)
               }
               atomic {
                   finished ? p
                   printf("finished Writer %d\n",p)
               }
               count = 100
```

`rdr_wrt_msg_v4.pml` (6/7)

```
...
      ::  count < 0 ->
              atomic {
                  finished ? p
                  printf("finished Writer %d\n",p)
              }
              count++
      od
    }

...
```

rdr_wrt_msg_v4.pml (7/7)

```
...
init {
    byte i

    atomic {
        for (i : 1 .. NRDRS+NWRTS) {  /* R1,R2,W3,R4,W5,R6,R7 */
            if
            ::  i == 3 || i == 5 ->
                    run ReaderWriter(i,writer)
            ::  else ->
                    run ReaderWriter(i,reader)
            fi
        }
        run Controller()
    }
}
```

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)

Invalid end state

```
...

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)

```
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++;
        }
    }
}
```