

Real-Time Programming

Summer Semester 08

Assignment #5

Homepage: http://www.iste.uni-stuttgart.de/ps/Lehre/SS2008/V_RTP/

Discussion of Solutions: July 2, 2008

1 Semaphores

Consider a shared data structure that can be both read from and written to. The readers and writers operate upon shared data that consists of two integer values, x and y . The data structure is in a consistent state if y 's value is double the value of x .

Show how semaphores can be used to allow *many* concurrent readers or a *single* writer but not both. Make sure that readers that access the shared data structure see it in a consistent state.

A semaphore implementation along with an unsynchronized version (i.e., with race conditions that cause inconsistent state) of the readers/writers is given at

http://www.iste.uni-stuttgart.de/ps/Lehre/SS2008/V_RTP/a5/rw.adb

You are asked to insert appropriate semaphore calls to eliminate the race conditions.

2 Parent/Child and Guardian/Dependant

For every task in the Ada program given at

http://www.iste.uni-stuttgart.de/ps/Lehre/SS2008/V_RTP/a5/guardians.adb

indicate its parent and guardian (or master in Ada terminology), and if appropriate its children and dependants. Also indicate the dependants of the **Guardians** and **Hierarchy** procedures.

Hint: The terms parent, child, guardian, and dependant are explained in Burns/Wellings, chapter 7.2, page 185.

3 Conditional Critical Regions

1. Rewrite the semaphore implementation of the readers/writers problem given at

http://www.iste.uni-stuttgart.de/ps/Lehre/SS2008/V_RTP/a5/rw.adb

using conditional critical regions instead of semaphores.

(Conditional critical regions are discussed in the lecture notes in section 7.4.1.)

Hint: The solution is much simpler and shorter than the semaphore implementation. It uses only two counters instead of four.

2. Compare conditional critical regions with monitors and Ada protected objects.

(Monitors and protected objects are discussed in the lecture notes in section 7.4.2 and 7.4.3, respectively.)

Briefly outline how the implementation of the readers/writers problem for monitors and Ada protected objects looks like.

4 Ada Protected Objects

Explain the synchronization behavior imposed by the following Ada protected object:

```
protected type Barrier is
  entry Wait;
private
  Avalanche : Boolean := False;
end Barrier;
```

```
protected body Barrier is
  entry Wait when Wait'Count = 42 or Avalanche is
  begin
    if Wait'Count = 0 then
      Avalanche := False;
    else
      Avalanche := True;
    end if;
  end Wait;
end Barrier;
```