

Re: Smart programming languages

Source: <http://coding.derkeiler.com/Archive/General/comp.programming/2005-02/0307.html>

From: Jim Rogers (jimmaureenrogers_at_att.net)

Date: 02/05/05

Date: Sat, 05 Feb 2005 00:47:04 GMT

Randy Howard <randyhoward@FOOverizonBAR.net> wrote in news:MPG.1c6db5faa88e3884989efc@news.verizon.net:

- >
- > *One thing that would be nice, but perhaps approaching impossible to*
- > *implement generically would be a special variable type for global*
- > *shared data values for counters, state, etc. that you want to*
- > *ALWAYS be locked via mutex or something similar for threaded code.*

Ada has provided this since 1995 through its protected objects.

Ada protected objects may have three kinds of operations.

Procedures automatically acquire an unconditional lock on the protected object.

Entries automatically acquire a conditional lock on the protected object.

Functions are read-only and automatically acquire a shared lock (shared with other functions and function calls) on the protected object.

I have explained this before in this group, and will explain it again using a bounded circular queue as an example.

Ada requires a separation of the interface definition from the implementation. The implementation is never visible to a caller.

The interface definition is also known as the specification.

The specification for my protected bounded circular queue is:

```
package Queue is
  type Queue_Index is mod 10;
  type Buffer_Type is array(Queue_Index) of Integer;
  protected Circular_Queue is
    entry enqueue(Item : in Integer);
    entry dequeue(Item : out Integer);
    function Length return Integer;
  private
    Buffer : Buffer_Type;
    Put_Index : Queue_Index := 0;
```

```
    Get_Index : Queue_Index := 0;  
    Population : Natural := 0;  
end Circular_Queue;  
end Queue;
```

The implementation is also known as the body.

```
package body Queue is  
  protected body Circular_Queue is  
    entry enqueue(Item : in Integer) when Population < 10 is  
      begin  
        Buffer(Put_Index) := Item;  
        Put_Index := Put_Index + 1;  
        Population := Population + 1;  
      end enqueue;  
    entry dequeue(Item : out Integer) when Population > 0 is  
      begin  
        Item := Buffer(Get_Index);  
        Get_Index := Get_Index + 1;  
        Population := Population - 1;  
      end dequeue;  
    function Length return Integer is  
      begin  
        return Population;  
      end Length;  
    end Circular_Queue;  
end Queue;
```

Some little notes for the above example:

A modular type, such as Queue_Index has a range modulo the number given in the type definition. In this example the range of values for this type is 0 through 9. All arithmetic on a modular type is modular. All addition and subtraction will result in a value in the range of 0 through 9. This modular arithmetic makes a modular type ideal for the index of a circular queue.

All protected entries have an automatically created and maintained queue. When a task (similar to a thread) calls a protected entry, and the boundary condition evaluates to FALSE, the call is placed in the entry queue. When any protected entry or procedure completes the boundary conditions for all entries for that protected object are re-evaluated. If an entry boundary condition is TRUE and there are one or more tasks in that entry queue, the entry is executed by the current task as a proxy for the queued task, and the queued task is allowed to proceed, receiving any value(s) passed out of the entry.

Procedures and entries use exclusive locks. Procedures have no associated queues, nor do they have any boundary conditions.

comp.programming: Re: Smart programming languages

Functions, because they are read-only, are allowed to have shared access (with other function calls) to the protected object. Functions implement a read/write lock.

The body of an entry can contain a requeue statement, allowing the current call to be transferred to another entry queue, or the same entry queue. This allows the creation of a chain of entry calls when needed.

Jim Rogers