Abstract

In this short paper two new pragmas are introduced and defined. Together they allow non-preemptive scheduling to be specified for Ada Programs. The syntax, rules and dynamic semantics for the scheme are described. Use is made of an additional execution resource – the execution-token.

1 Introduction

At the last two IRTAWs[1, 2] consideration was given to a non-preemptive version of Ravenscar, and non-preemptive execution in general. Class A (or Class 1) software (as defined in safety standards such as DO-178B [4]) typically has a very restricted architecture. Often only periodic behaviours need be supported. In order to reduce non-determinism and to increase the effectiveness of testing, non-preemptive execution is desirable[3]. Although non-preemption can reduce schedulability, it can be analysed and there are ways of improving its effectiveness (by restricting the maximum length of any non-preemptive section of code).

The ARM allows new scheduling policies to be defined and additional execution resources to be used in these policies. This short paper defines such a policy for non-preemptive execution. Note that non-preemptive behaviour does not preclude interrupts either for the run-time (to manage the delay queue) or for application-level interrupt handlers.

2 Definition

The style of the Ada LRM is used to introduce these new features.

2.1 Syntax

\[
\text{pragma Task_Dispatching_Policy (Non_Preemptive_Fifo_Within_Priorities);} \\
\text{pragma Locking_Policy (Non_Preemptive_Locking);} 
\]

2.2 Post-Compilation Rules

If the Non_Preemptive_Locking policy is specified for a partition then Non_Preemptive_Fifo_Within_Priorities shall also be specified for that partition.

2.3 Dynamic Semantics

Under D.2.1.9 we define an additional execution (non-preemptive) resource, the execution-token. Each processor has one execution-token resource. A ready task must acquire the execution-token before it can become the running task. When the Non_Preemptive_Fifo_Within_Priorities policy is in effect the modification to the ready queues are identical to the existing preemptive policy Fifo_Within_Priorities.

The running task releases the execution-token whenever it becomes blocked. It also releases the execution-token whenever it executes a delay statement (whether this results in blocking or not).

A new running task is selected and is assigned the execution-token whenever the previously running task becomes blocked or releases the execution-token. The rule for selecting the new running task follows the policy of Fifo_Within_Priorities.

The locking policy, Non_Preemptive_Locking is defined as follows:

- if the protected object contains either an Interrupt_Priority pragma, an Interrupt_Handler or Attach_Handler then the rules
defined for locking policy Ceiling_Locking
apply;

- if none of the above pragmas are present then no run-
time code need be generated to protect the object, in
particular the priority of the calling task need not be
changed;

- pragma Priority must not be present in any pro-
tected object.

NOTE:

1. The running task may release the execution-token, by
executing, delay 0.0 but be reassigned it immediately
if it is at the head of highest priority ready queue.

2. Implementation Permission 9.5.3 (22) still applies.

3. It remains a bounded error to call a potentially block-
ing operation from within a PO.

3 Conclusion

A definition of non-preemptive scheduling has been given.
This could be used with the full Ada tasking facilities or
with a restricted subset such as Ravenscar. Non-preemptive
execution can deduce schedulability if a low priority task
has a long execution time. However, this can be countered
by restricting the maximum length of any non-preemptive
section of code. Thereby providing a simple means of in-
creasing schedulability – the judicious introduction of delay
0.0 statements.

The most restricted subset of language features comes
from using non-preemption, the Ravenscar subset and by
also disallowing application interrupts. Although this is a
very static approach, it has a number of advantages over the
conventional use of a cyclic executive. The primary advan-
tage is that it allows unrelated iteration rates for the tasks to
be supported.

References

summary: The Ravenscar profile and implementation
issues. In A. Burns, editor, Proceedings of the 9th In-
ternational Real-Time Ada Workshop, volume XIX(2),

profiles. In A.J. Wellings, editor, Proceedings of the

In A.J. Wellings, editor, Proceedings of the 8th Inter-
national Real-Time Ada Workshop, pages 27–32. ACM

[4] Software Considerations in Airborne Systems and
Equipment Certification DO-178B/ED-12B. RTCA,