GEHANI84
Gehani, N.H., Cargill, T.A.;
Concurrent Programmin in the Ada Language: The Polling Bias;
Software Practice and Experience 14, 5 (May 1984).
Based on their experience with concurrent programming in Ada, the authors have found several deficiencies in the language facilities provided to implement the rendezvous mechanism. (Ada uses the rendezvous as its principle means of task synchronization and communication.) In particular they have found that there is a tendency to produce tasks that poll one another. This is generally results in inefficient programs and can usually be avoided by careful program design. The paper also suggests several modifications to Ada that would reduce these tendencies.

Two forms of polling are identified: rendezvous polling and information polling.

Rendezvous polling refers to the situation where one task is trying to rendezvous with another but does not suspend itself if the rendezvous cannot take place immediately. Between attempts to rendezvous the task may or may not deo something useful . If no useful work is done then the task is essentially in a busy-wait loop, using system resources without accomplishing anything.

Information polling occurs when one task repeatedly rendezvous with another task until it gets the result that it wants. For example, task A requests a resource controlled by task B and is refused. If task A then repeatedly requests the resource, instead of doing something else, task A is said to be infomation polling task B.

While information polling is typically the result of poor program design, rendezvous polling is the result of a mixture of poor program design and inadequate language facilities.

The authors identify three major problem areas.
The Conditional Entry Call - The ’else’ clause in the programs. This is aggravated by the restriction that only one entry call may appear. (ie. no alternatives as in the selective wait statement).

Handling an Entry Family - The absence of a language facility for specifying a family of entries in one "accept" statement usually results in rendezvous polling as the task repeatedly loops through all possible entries, one at a time, checking for a rendezvous.

Restrictions on the Selective Wait Statement - Conditional Entry Call, the ’else’ clause in the selective wait statement is the source of the problem. The authors feel that the tendency to write tasks that poll through a selective wait statement could be reduced by :

allowing non-tasking statements as a (possibly guarded) select alternative.

allowing an entry call as a selective wait alternative.

While the authors propose several language changes that might reduce the

polling bias, they realistically recognize that Ada willnot be changed in the near future. Therefore, they have also provided some guidelines to help in the design of non-polling programs.

The article presents a specialized topic in an accessible manner, including a brief tutorial on the Ada language facilities for task synchronization and communication. The authors have taken a very practical stance by providing both proposals for improving Ada and program design guidelines for avoiding problems. The article woul be of interest to people writing concurrent Ada programs as well as anyone designing a concurrent language that uses the rendezvous mechanism.