Sequential Pascal with Recovery Blocks

S. K. SHRIVASTAVA

Summary. The programming language Sequential Pascal has been extended to include recovery blocks. This paper describes the modifications made to the kernel and interpreter of Brinch Hansen's Pascal system to support recovery blocks and the associated recovery caches needed for state restoration.

Key Words: Sequential Pascal  Recovery blocks  Recovery cache  Fault-tolerant software

Introduction

A program structure called recovery block has been proposed in the literature as a means of constructing fault-tolerant software [1, 2] (defined to be software that produces acceptable results despite faults in the hardware and software). This paper describes an implementation of recovery blocks using Sequential Pascal [3] as the host language. The objectives of this paper are twofold: firstly, the implementation details are believed to be sufficiently interesting in their own right and, secondly, recovery blocks have attracted wide attention (for example, they are actively being evaluated for aerospace applications [4]); thus, an account of a method of inclusion in Sequential Pascal, a language that is widely used for research in programming methodology, should prove interesting to workers in the field of fault-tolerant programming. The paper also demonstrates that the inclusion of recovery blocks into existing programming systems can be a practical proposition.

Recovery blocks were first implemented by my colleagues [5], one of the aims of their work was to investigate a suitable computer architecture for directly supporting recovery blocks. The resulting system could however support only relatively simple sequential programs. A second experiment was therefore started with the aim of developing a system capable of supporting realistic sequential and concurrent programs incorporating recovery blocks. This paper describes the first phase of this experiment – the development of a system that is capable of supporting realistic sequential programs with recovery blocks. Work is underway to extend this system to support the features necessary for fault-tolerant concurrent programming [6]. While recovery blocks are described briefly in the next section, a familiarity with the concepts presented elsewhere [1, 2] would be helpful to the reader.

Recovery Blocks

The syntax as incorporated in Sequential Pascal is as shown. The acceptance test (a Boolean expression) is evaluated after the execution of the primary.
ENSURE ⟨acceptance test⟩ BY
     ⟨statement⟩ “primary”
ELSE-BY ⟨statement⟩ “first alternative”
...
ELSE-BY ⟨statement⟩ “nth alternative”
ELSE-ERROR;
...

If the result is true, the statement following the recovery block is executed. However, if the result is false, the state of the computation is restored to that at entry to the recovery block and the first alternative is tried and so on. If all the alternatives fail to produce acceptable results, then this is regarded as a failure of the entire recovery block — any recovery actions must be undertaken by the enclosing recovery block, if any (recovery blocks may be nested). A 'recovery cache' is used for recording the state of the computation and restoring it when the primary or the current alternative fails. The recovery cache is organized as a stack and contains recovery data for the recovery blocks entered but not yet exited. The recovery data consist of the addresses and the prior values of the global variables updated inside a given recovery block, so that the act of state restoration merely consists of copying the prior values into the variables. When an acceptance test is passed, some of the recovery data of this recovery block may have to be merged with the recovery data of the enclosing recovery block (if any). Precise details of this merging and other related aspects of recovery cache are discussed elsewhere [1, 5].

The Pascal System

The Pascal System, as developed by Brinch Hansen's group [11], is capable of supporting a number of concurrent processes programmed in Concurrent Pascal [7]. A process is capable of executing sequential programs written in Sequential Pascal (this language is closely similar to Pascal [8], from which it has been derived).

A process can make available some of its procedures to the sequential program it is running — this forms the basis of the interface between user programs (written in Sequential Pascal) and the operating system (written in Concurrent Pascal). Such procedures have been called prefix procedures (as a consequence of prefix procedures, no input—output has been defined for Sequential Pascal; rather, a system designer can program appropriate input—output procedures in Concurrent Pascal as prefix procedures). Both the concurrent program and sequential programs are executed interpretively by a simple stack machine programmed to run on the host hardware (PDP 11/45). Certain details of this interpreter and related programs are of interest within the context of this paper.

The Kernel and Interpreter

The kernel is the initial piece of software written to run on the base machine and it implements processes, synchronizing primitives, queues, basic input and