Exception handling

John Reid, JKR Associates and
Rutherford Appleton Laboratory

BCS Fortran Specialist Group
London, 28 September 2017

Abstract

Exception handling was included in 1994 drafts of Fortran 95, but removed because it was not ready in time.

It is a candidate for Fortran 2020.

I will describe the 1994 facility and discuss its adaptation as an extension of Fortran 2015.

For a full description, see 94-258r4 on the WG5 web site **https://wg5-fortran.org/**

Enable construct

Here is a simple example of the enable construct

```
enable (overflow, divide_by_zero)
   :  ! Try a fast algorithm for inverting a matrix.
handle
   :  ! Fast algorithm failed; use slow one.
end enable
```

Avoids very defensive coding unless it is really needed.

Conditions

The conditions have the scope of intrinsics and are:

allocation_error, **deallocation_error**, **insufficient_storage**

bound_error, shape, many_one, not_present, undefined

io_error, end_of_file, end_of_record

overflow, underflow, divide_by_zero, inexact, invalid

integer_overflow, integer_divide_by_zero

intrinsic

system_error

Signalling outside enable blocks

insufficient_storage signals outside an enable block.

For each other condition, it is processor dependent whether it signals outside an enable block.

Combined conditions

storage is equivalent to **allocation_error, deallocation_error, insufficient_storage**

IO is equivalent to a list of all IO conditions

floating is equivalent to **overflow, divide_by_zero, invalid**

integer is equivalent to listing the two integer conditions.

usual is equivalent to **storage, IO, floating, intrinsic**

all_conditions is equivalent to listing all the conditions.

Enable construct

The enable construct has the general form

enable *[(list)]* *[***,immediate***(list)]***

[enable block]

[**handle** *[(list)]*

handle block]

end enable

Imprecise transfer

If one of the enabled conditions signals, there is an imprecise transfer to the handle block – any variable that might be defined or undefined in the enable block becomes undefined.

This allows code motions for optimization.

enable, handle, and **end enable** statements provide barriers to code motion.

Nesting

Enable constructs can be nested.

If an enable statement is executed and a condition that is enabled or about to be enabled is signalling, there is a transfer of control to the next outer handler for that condition or **return** (**stop** in main program) if there is no such handler.

Hence conditions enabled in an enable block always start quiet. They are set quiet at end of handle block.

Conditions enabled in an outer block remain enabled.

Enable construct with no handle

If an enabled condition signals and there is no handle block, control is transferred to the next outer handler for the condition. Allows transfers to be more precise, e.g.,

```
enable (overflow, divide_by_zero)
   do k = 1,n
      enable
         :
      end enable
   end do
handle
  :  ! We know that k-1 steps have executed normally
end enable
```

Immediate

A statement that might signal an **immediate** condition is treated as being followed by

enable

end enable

If the condition signals, there is an immediate transfer to its handler.

Branching

Branching out of an enable construct is not permitted, including a **cycle** or **exit** statement for an outer construct. This limits the uncertainty of what has been executed when the handler is entered.

return is permitted. Any signalling conditions continue to signal.

Inquire statement

The condition-inquire statement has the form

condition_inquire (*condition***,** *integer-variable***)**

or

condition_inquire (*character-array***)**

Returns the value of the condition or the names of all the signalling conditions.

Quiet condition has the value 0.

Signal statement

The signal statement has the form

signal (*condition***,** *integer-expression***)**

It causes the condition to signal or become quiet.

Adaptation to Fortran 2015

Better scoping with **condition_type** in an intrinsic module.

Allow an array to appear in condition list on an enable statement.