Common Fortran Error messages

Common Fortran Error messages

When you execute your program, you may encounter run-time errors that are difficult to trace. The three most common errors are segmentation violations, bus errors, and arithmetic exceptions.

Segmentation violations

A segmentation violation occurs when some part of your code attempts to access a part of memory that is not defined by your program. A common cause of this problem is attempting to access an array with an invalid subscript. Arrays in Fortran must have integer subscripts whose values are between 1 and the
dimension of the array the subscript refers to. For example, if you declared an array as:

```
real foo(10,5)
```

and attempted to access element foo(32,3), a segmentation violation would occur because the value 32 is greater than the dimension of the array.

Bus errors

A bus error occurs if the data sets passed during a main/subprogram to subprogram interface are not of equal size. One possible cause of this problem is an unequal number of arguments in the argument list of a 'call' statement and the corresponding subroutine declaration as in:

```
call foobar(x,y,z)   <-- 3 arguments

 subroutine foobar(a,b,c,d)  <-- 4 arguments!
```

Another possible cause is an argument list in a call statement that does not contain the same variable types as the subroutine declaration. For example, the code below would result in a bus error because array 'foo' is declared as real, and array 'dummy' is declared as double precision.

```
real foo(10,5)
 integer i, j
 call foobar(foo,i,j)   <-- real,int,int
            |
            |==> these don't match!
            |
 subroutine foobar(dummy,idummy,jdummy)  <-- double,int,int
 double precision dummy(10,5)
 integer idummy, jdummy
```

Arithmetic/floating point exception

An arithmetic or floating point exception (also labelled as "TRACE/BPT Error") occurs when the value of a variable exceeds its proper range in some way. This error commonly occurs when one attempts to divide by zero. Since an infinite value cannot be represented by the computer, an error occurs. For example, the code:

```
integer i,j,k
 i = 1
 j = 0
 k = i/j
```

will generate the error "Arithmetic Exception: Integer divide by 0" because an attempt to divide by zero (assign k to infinity) was made.

An arithmetic exception can also occur if the value of a variable exceeds the largest value that can be represented by the corresponding data type. For example, the code:

```
real foo
 read*, foo
 stop
 end
```

would generate the error "Arithmetic Exception: Floating point overflow" if the value 1+E1000 (10 to the 1000 power) were entered for foo. This is because 1+E1000 is greater than the value that can be represented by a real variable.

Unfortunately, when a program encounters one of the three errors described above, no reference is made to the source of the problem in your code. If you are unable to locate the offending source code, you should use the dbx debugger. It provides a quick and easy way to find such errors in your source code.