Using the Sun Debugger dbx

1. Overview

Very often your programs may contain mistakes (bugs) which cause them to fail or give incorrect results. **Finding your bug is a process of confirming the many things you believe are true, until you find one which is not true.** This can be time consuming. One way to find out what is happening in your code is to write out variables at certain points in your code and see if those variables contain the values you think they should have.

A debugger is a software utility which lets you follow the code during execution, stops at specified points and lets you inspect the variables.

The debugger of the Sun Studio 12 is called **dbx**. The manual page is located at *http://docs.sun.com/app/docs/doc/819-5257*.

dbx can do mainly four types of things to help you catch bugs in the act:

- Start your program, specifying anything that might affect its behavior.
- Make your program stop on specified conditions.
- Examine what has happened when your program stopped.
- Change things in your program so that you can experiment with correcting the effects of one bug and go on to learn about another.

The program being debugged can be written in any of the languages the Sun Studio 12 supports. In our case this will be Fortran 90.

2. Preparations

Your code needs to be compiled in special way so that the executable is prepared for dbx. Let us take the program example.f90. Compiled with
f95 -g -o example example.f90
produces an object file (executable) 'example' which contains a lot of extra information needed by the debugger. So, without the **-g** option dbx is essentially useless, but for normal execution you would not want to use the **-g** option, since it makes execution slow.

3. Running dbx

To start **dbx** type

dbx example

Once **dbx** is running, you should see the *(dbx)* prompt. At this point you can start issuing **dbx** commands. You exit again by typing **quit** at the *(dbx)* prompt.

4. dbx commands

There are many **dbx** commands described in the manual. Some of the basic commands are *run, stop, next, cont, print* which will get you started.

run *[arguments]* example
Begin executing the object file example, passing optional command-line *arguments*. The arguments can include input or output redirection to named files, as in a normal command line.

run will execute the object file, and if crashes will tell you where. If you want to examine certain section in your code, you need to stop the execution, i.e. set break points. This can be done several ways.

stop in *subnam*
where *subnam* is e.g. the name of a subroutine or function. **stop in** MAIN will stop at the first executable statement of your main program. (Hint: It will be useful if you have an output or a editor with line numbers, since **dbx** will refer to the line numbers of your program.)

stop at n
will put a stop in the main program at the line n

stop at *"subroutine.f90":n* will put a stop in the subroutine at line n.

When you **run** the debugger will stop at the first breakpoint you set. Once at the first break, you can examine all portions of the code which have been executed up to there.

print *expression* will print the current value of the variable *expression*, which can be single variables or arrays. To print values of two-dimensional Fortran array elements use the format *print array[1,2].*

next will execute the next line

cont will continue execution up to the next breakpoint.

list *[n1 [,n2]]* or *func*
lists the source text between lines $n1$ and $n2$, or on lines surrounding the first statement of *func*. With no arguments, **list** lists the next ten lines from the current active line.

5. Specific example

You may want to start by using **dbx** to set some break points within your code. To step through your code at the very beginning, you need to stop in the main routine.

(dbx) stop in MAIN

Now you can issue the **run** command to start execution of your object file. You will get the process id and the name of the object file being executed.

(dbx) run

At this point, you may use the **list** command for the first 10-line listing of your source code:

(dbx) list

Use the **stop** command to set break-points at various lines or procedures within the object-file:

(dbx) stop at 10
(dbx) stop at sub123.f:10
(dbx) stop in sub123
(dbx) stop in sub456 if i == 24

The execution will stop in the example at line 10, or at line 10 in subroutine *sub123*, or in the subroutine *sub123*, or in the subroutine *sub456* when i is equal to 24. To continue execution at any point in your debugging session, issue the **cont** command.

(dbx) cont

To restart your debugging session, issue the **rerun** command:

(dbx) rerun

To catch the place where floating point exceptions occured, enter at the beginning of the dbx run **catch FPE**.

To exit **dbx**, type **quit**

(dbx) quit

6. Catch Floating Point Exceptions

Compile code with

f95 -g -ftrap=common example example.f90

then start dbx as usual with
dbx example

before running, issue command
(dbx) **catch FPE**
(dbx) run example
and the debugger will give the line in the code at which the floating point exception occured.