CERES Software Bulletin 95-12

Fortran 90 Linking Experiences, September 5, 1995

1.0 Purpose:
To disseminate experience gained in the process of linking Fortran 90 software with library routines compiled under Fortran 77 compiler.

2.0 Originator:
Lyle Ziegelmiller (lyle@larc.nasa.gov)

3.0 Description:
One of the summer students, Julia Barsie, was working with a plot program which was written in f77. This program called routines from a graphics package known as NCAR, which is also written in f77. Everything was fine.

The plot program was converted to f90, and a new version of the NCAR graphical package was released, which was written in f77. A problem arose when trying to link the new f90 version of the plot program with the new f77 release of NCAR; many undefined references were reported by the linker. This bulletin is intended to convey what was learned in the effort to accomplish this linking.

The first step I took was to issue the "-dryrun" directive to the f77 compiler when using it to compile the original f77 plot program and the original NCAR graphics library. "-dryrun" causes the linker to produce an output detailing all the various libraries that it links with. Note that these libaries are in addition to the libaries you would select on the command line. For example, you might compile a program with erbelib, but the linker would have to link with librarie(s) that contain the definitions of sine or cosine. Anyway, it was my hypothesis that if everything compiled and linked with f77, then all the libraries must be contained in the output from the f77's "-dryrun" command. So I cut and pasted all these libraries to the end of the f90 compile command. This should have worked, but of course, it didn't. I was linking f77 libraries with the f90 compiler/linker. Some of the routines were defined more than once by this first hack attempt. The f90 linker didn't like all the f77 libraries, and erroneously reported syntax errors in the f77 libraries. I then began commenting out the offending libraries. This was an iterative process, because (it seems), once the linker encounters something it doesn't like, it complains and then gives up, just as I occassionally did. After beating on it for a couple weeks, I got the list of undefined references down to four routines. So the question was, where in the world are these four routines. If you issue the command:

find /opt/optional -name '*.a' -print

you will get an extensive list of libraries from which the linker may choose. Save this output to a file, called tmp1, then, issue the following command. This command assumes

you are missing the routines s_wsle, s_copy, c_conv.

```
foreach i ( `cat tmp1` )
  echo $i
  ar t $i | awk '/s_wsle|s_copy|c_conv/'
end
```

This locates the libaries containing these routines. Now, add the libraries in which they are found to your f90 compile command, and try again. It is possible that the routines will still be unfindable by the linker. I don't know why. If this happens, extract the object code for the offending routines from the library, put them in the directory in which you are trying to compile, add their names to the compile command, and try again. In my case, this is what finally worked. Someone told me that removing all the routines from a library and restoring them can help. That doesn't make a lot of sense either, but I guess it works.

Below is the script which I used to finally get the plot code to compile. I present it here to demonstrate the use of the "\" to make things more readable. Note how, for example, r_atn2.o appears explicitly on the command line. This is the sort of thing that should be refereced indirectly by specifying a library.

```
#!/bin/csh
#
f90\
-Qpath\
/opt/net/lib/f90\
-o\
ssfplot\
-lm\
f90_kind.f90\
ssf_typdef.f90\
tk_replace.f90\
plot.f90\
user.f90\
io.o\
-lF77\
pow_ri.o\
r_atn2.o\
r_mod.o\
s_stop.o\
/opt/optional/Fortran2.01/SC2.0.1/values-Xs.o\
/opt/optional/Fortran2.01/SC2.0.1/cg87/_crt1.o\
/opt/optional/Fortran2.01/SC2.0.1/libansi.a\
/opt/optional/nlib/libncarg.a\
/opt/optional/nlib/libncarg_gks.a\
/opt/optional/nlib/libncarg_loc.a
```

Well, that's as far as I've gotten so far. Let me know if you have any questions. I guess my

parting words are to do everything explicitly. Don't count on some environement variable like LD_LIBRARY_PATH. I like to be able to see everything that's going on.