my Fortran web page

Nasser M. Abbasi

July 2, 2015 page compiled on July 2, 2015 at 2:25am

Contents

1 using OPENGL with Fortran

1.1 source code

This zip file contains the source code and examples and scripts needed to build client Fortran program using the Fortran 2003 openGL binding

`nma_fortran_opengl.zip`

The above is a slight modification based on the original binding located at

http://www-stone.ch.cam.ac.uk/pub/f03gl/index.xhtml

based on work by Anthony Stone, Aleksandar Donev and the original f90 open GL binding by William F. Mitchell.

What I did is document the F2003 openGL binding with diagrams below, made small code changes to make build with F2008 and build the examples in it as static to make it easier for someone to download and run without having to have openGL installed on their Linux. This was done for Linux only not for windows. Below is more description of my changes to the binding.

The executables of the examples are in the above zip file. I added .exe to the names just to make it more clear. These are linux executables, not windows.

1.2 description

This describes how to use 2003 opengl with Fortran and few examples I wrote using this API as I learning openGL. First a description of the build structure is given. This is the description based on my own layout of the official binding obtained from http://www-stone.ch.cam.ac.uk/pub/f03gl/index.xhtml

I have made small changes to the file naming (mixed case changed to all lower case) since it is less confusing this way, as Fortran is case insensitive and now the file names and the module names all match.

In addition, I broke one module file into 2 files so that there is 1-1 correspondence with file name and module name.

Some code changes are made to the binding files to make them build with std=f2008.

The description below reflects what I have in the above zip file not the original and official Fortran 2003 openGL binding.

There are now 4 Fortran files that make up the Fortran 2003 binding, and one C file. They are

1. opengl_kinds.f90.txt generates opengl_kinds.mod, all other files USE this one.
2. opengl_glu.f90.txt generates opengl_glu.mod
3. opengl_gl.f90.txt generates opengl_gl.mod
4. opengl_freeglut.f90.txt note: this generates opengl_glut.mod and not opengl_freeglut.mod
5. glut_fonts.c.txt generates glut_fonts.o only

This picture below shows the layout of the binding tree as I have it now (again, this is slightly different naming from the official tar file)

```
 $HOME/
   |
   |
  fortran_binding/
   |
   |
opengl_freeglut.f90 (USE opengl_kinds)  ---> opengl_freeglut.o
                                             opengl_glut.mod

opengl_gl.f90 (USE opengl_kinds)        ---> opengl_gl.o
                                             opengl_gl.mod

opengl_glu.f90 (USE opengl_kinds)       ---> opengl_glu.o
                                             opengl_glu.mod

opengl_kinds.f90                        ---> opengl_kinds.o
                                             opengl_kinds.mod

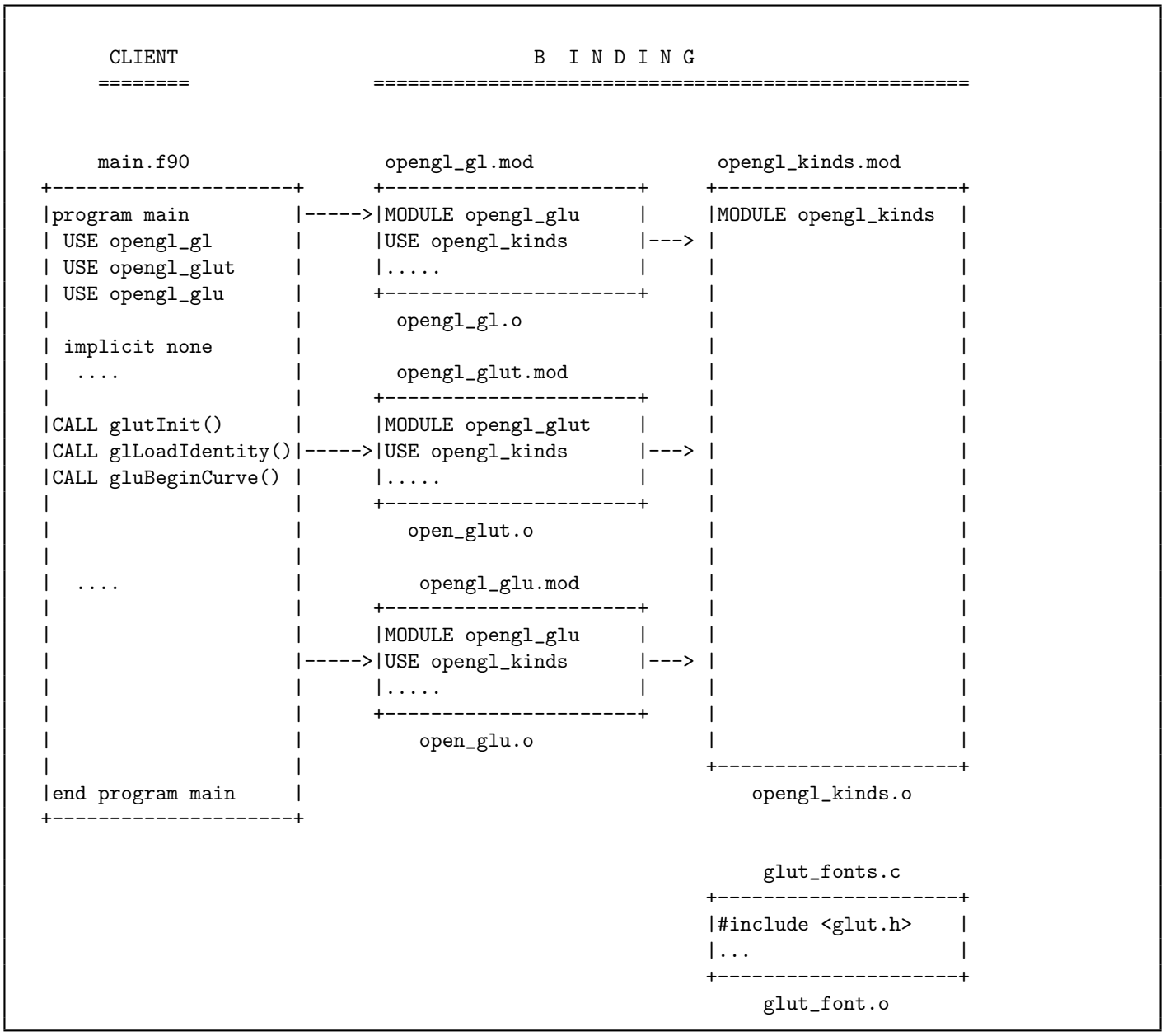
glut_fonts.c --> glut_fonts.o
```

A client program needs to link against the 5 object files

```
opengl_glut.o, opengl_gl.o, opengl_glu.o, opengl_kinds.o, glut_fonts.o
```

and the client will contain `USE opengl_gl`, `USE opengl_glut` and `USE opengl_glu` in the source to be able to access the functions provided by these bindings.

Therefore, a typical structure of a Fortran program that uses opengl will be as follows



In the above diagram, the client program `main.f90` needs to link to the above 4 object files generated from the 4 Fortran files, and against the object file generated from the one C file `glut_fonts.o`

The above shows the object files that needs to be linked against. In addition, one needs to link to the opengl and glut and X libraries of course. For static linking, other libraries are needed. Diagram below shows all libraries needed, and a script is including below to do that.

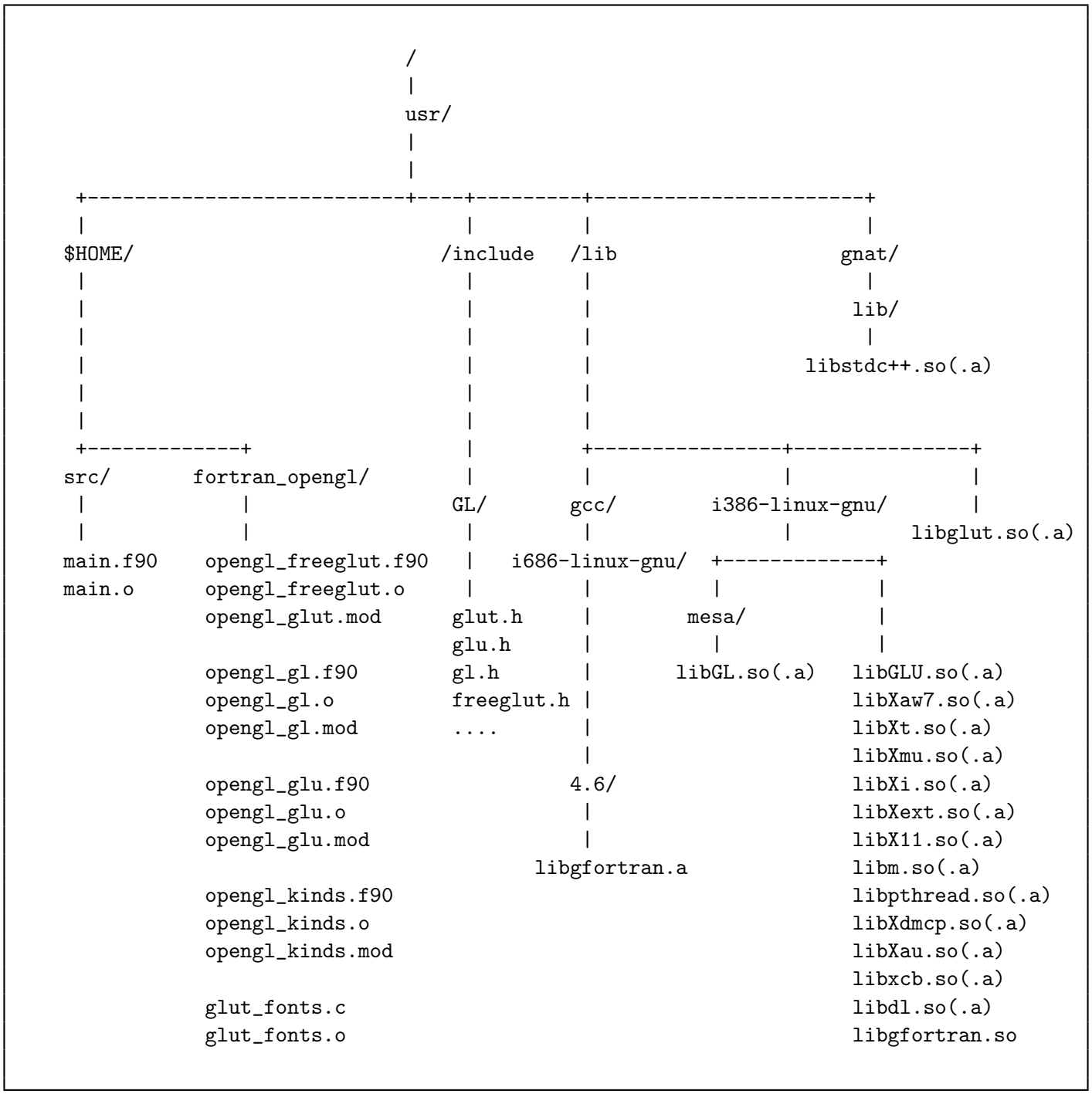
The diagram below shows everything that is needed to compile and link against.

This assumes a Linux 32 bit system. This is based on Linux mint 32 bit. On different Linux distributions or on 64 bit, the tree will can and will look a little different depending on the distribution.

To compile the C file `glut_fonts.c` the file `GL/glut.h` is needed.

In the diagram below it is assumed that `main.f90` is located in your own folder, say `$HOME/src` and that the Fortran opengl binding files are located in `$HOME/fortran_opengl`.

The following diagram shows all the files needed to build an executable as shared or static. This is the layout on Linux 32 bit (Linux Mint 13)



1.3 building the Fortran OPENGL binding

Building the openGL binding only is simple. It is just compiling the above 5 files to generate the 5 object files and the 4 .mod Fortran files. The following bash script accomplishes this

`build_fortran_opengl_binding.sh.txt`

I did not use Makefiles here in order to make it easier. Any one who is familiar with Makefiles can easily use these scripts to make a Makefile from them.

To building the binding, change to the directory where the Fortran OPENGL binding is located and execute the build bash script

```
./build_fortran_opengl_binding.sh
```

Here is the listing of the above script

```
#!/bin/bash

#the compile flags
FFLAGS="-Wall -Wextra -pedantic -fcheck=all
        -fcheck=do -fwhole-file -funroll-loops -ftree-vectorize
        -Wsurprising -Wconversion-extra"

#compile the fortran files
gfortran -std=f2008 $FFLAGS -c opengl_kinds.f90  #must be first one
gfortran -std=f2008 $FFLAGS -c opengl_freeglut.f90
gfortran -std=f2008 $FFLAGS -c opengl_glu.f90
gfortran -std=f2008 $FFLAGS -c opengl_gl.f90

gcc  -Wall -Wextra -pedantic  -I/usr/include -c glut_fonts.c
```

1.4 building a client fortran program using the OPENGL binding

After building the Fortran OPENGL binding as shown above, now you can build a client program that uses it.

Two ways to build a client program are given. One is a dynamic build, which builds against the shared opengl and X libraries (.so), then a static build which uses the .a libraries.

The static build would be better if you want to send the executable to someone else to run the program on their computer which might not have all the libraries installed or different versions.

1.4.1 dynamic build

With all the above in place, the following script builds a client program `simple3.f90` program that uses Fortran openGL

`build_dynamic_opengl_client_linux_mint.sh.txt`

Change to the directory where you have your client Fortran program and type

```
./build_dynamic_opengl_client_linux_mint.sh
```

Now run the program generated from the above script

```
./simple3
```

If you get error like this

```
OpenGL Warning: XGetVisualInfo returned 0 visuals
```

then try the following command `LIBGL_ALWAYS_INDIRECT=1  ./simple3`

1.4.2 static build

To build static program, all what is needed is to link statically against the .a libraries instead of the shared libraries. But now the order of libraries is important.

The following script accomplishes this

`build_static_opengl_client_linux_mint.sh.txt`

Change to the directory where you have your client Fortran program and type

`./build_static_opengl_client_linux_mint.sh`

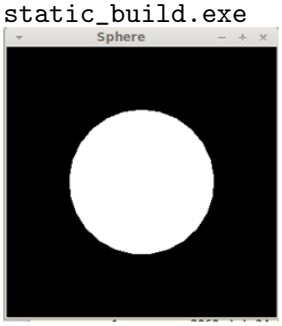
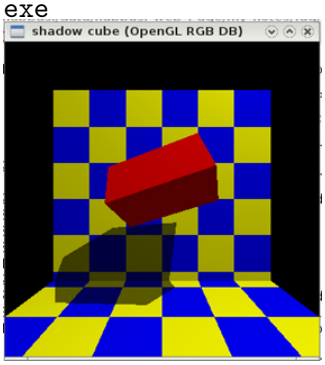
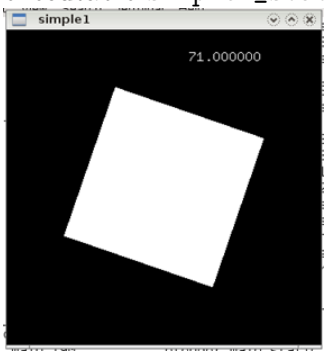
1.5 Examples of statically prebuild Fortran programs build with openGL binding

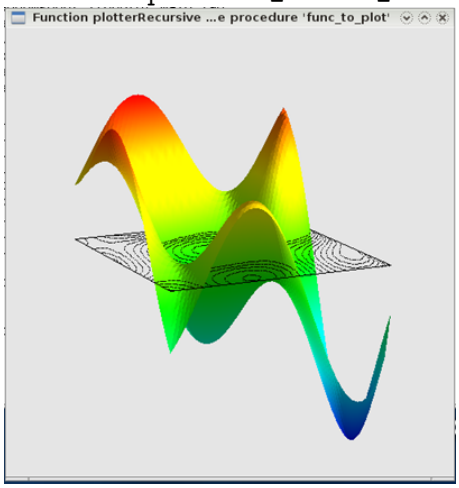
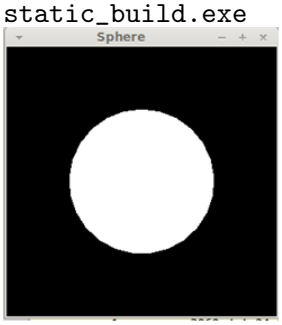
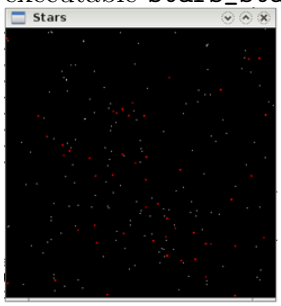
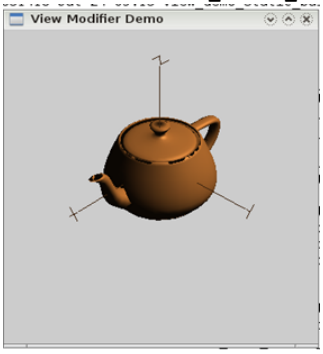
This table includes the examples from the official openGL fortran 2003 binding build statically on Linux. Each example has a screen shot and link to the source code and the executable. You can download the executable to Linux and run it. Hopefully it will run as is since it is statically linked.

I added .exe extension to the executable since some browsers have a bug in them where the file will be corrupted during download if it is binary but does not have .exe extension. This was the case using Firefox on linux.

To download the .exe, do SAVE AS and save it on your Linux file system, then run it as you would run any executable on linux.

All of these executables are in the above zip file. So if you downloaded the zip file, these are in there already.

`RandomSphere_FreeGLUT.f90` executable `randomspher_freeglut_main_static_build.exe` 	scube.f90 `scube_mod.f90` executable `scube_static_build.exe` 
simple2.f90 executable `simple2_static_build.exe` 	`blender_main.f90` blender.f90 executable `blender_main_static_build.exe` 

plotfunc.f90 function_plotter.f90 modview.f90 executable plotfunc_static_build.exe 	randomspher_freeglut_main.f90 RandomSphere_FreeGLUT.f90 executable randomspher_freeglut_main_static_build.exe 
stars.f90 stars_mod.f90 executable stars_static_build.exe 	view_demo.f90 view_demo_callbacks.f90 view_modifier.f90 executable view_demo_static_build.exe 

2 small examples of Fortran code

All these examples are build with this Makefile

2.1 f01.f90

f01.f90

```
!-- matrix transpose in Fortran
!-- Nasser M. Abbasi Feb 12, 2012
!--
!-- gfortran -std=f2008 f01.f90

program f01
  implicit none
  integer :: B(2,3),A(3,2);

  B(1,:) = [1,2,3]
  B(2,:) = [3,4,5]
```

```
  CALL print_matrix(B)

  A = transpose(B)

  CALL print_matrix(A)

CONTAINS
  subroutine print_matrix(A)
    implicit none
    integer, intent(in) :: A(:,:)
    integer :: i

    PRINT *,'--------------------'
    DO i = 1,size(A,1)
       print '(3i3)', A(i,:)
    END DO

  end subroutine print_matrix
end program f01

$./a.out
 --------------------
  1  2  3
  3  4  5
 --------------------
  1  3
  2  4
  3  5
```

2.2 f02.f90

f02.f90

```
!-- matrix, matrix multiply
!-- Nasser M. Abbasi Feb 12, 2012
!--
!-- gfortran -std=f2008 f02.f90

program f02
  implicit none
  integer :: A(2,3)

  A(1,:) = [1,2,3]
  A(2,:) = [3,4,5]

  CALL print_matrix( matmul(A, transpose(A)) )

CONTAINS
  subroutine print_matrix(A)
    implicit none
    integer, intent(in) :: A(:,:)
    integer :: i

    DO i = 1,size(A,1)
       print '(3i3)', A(i,:)
    END DO
```

```
  end subroutine print_matrix
end program f02

$gfortran -std=f2008 f02.f90
$./a.out
 14 26
 26 50
```

2.3 f03.f90

f03.f90

```
Translate this Matlab line to Fortran:

exp((0:1/100:1)*pi*2*sqrt(-1))

PROGRAM f03
IMPLICIT none
INTEGER, PARAMETER :: dp = SELECTED_REAL_KIND(15)
COMPLEX, PARAMETER :: arg=CMPLX(0_dp,4.0_dp*ACOS(0.0_dp),KIND=KIND(dp))
INTEGER, PARAMETER :: N=5
INTEGER :: i
REAL(KIND=dp), PARAMETER :: x(N+1) = (/ (DBLE(i)/DBLE(N), i=0,N)/)


   DO i=1,SIZE(x)
     PRINT*, exp(x(i)*arg)
   END DO

END PROGRAM f03

>gfortran   -fcheck=all -Wall -Wconversion -Wextra -Wconversion-extra -pedantic f11.f90
>./a.out
 (  1.0000000000000000     ,  0.0000000000000000     )
 ( 0.30901696111734539     , 0.95105652710120292     )
 (-0.80901703548360204     , 0.58778519571125987     )
 (-0.80901693271195796     ,-0.58778533716428760     )
 ( 0.30901712740535175     ,-0.95105647307094476     )
 ( 0.99999999999998468     ,  1.7484556000744883E-007)

Matlab

>> exp((0:1/5:1)*pi*2*sqrt(-1)).'

  1.000000000000000 + 0.000000000000000i
  0.309016994374947 + 0.951056516295154i
 -0.809016994374947 + 0.587785252292473i
 -0.809016994374947 - 0.587785252292473i
  0.309016994374947 - 0.951056516295154i
  1.000000000000000 - 0.000000000000000i
```

2.4 f04.f90 with mod file

f04.f90 `foo_mod.f90`

```
--- file  foo_mod.f90 ---------

module foo_mod
!-- Overload 2 subroutine names
  implicit none
  interface foo
    procedure foo_i,foo_c
  end interface
contains
   subroutine foo_i(arg)
     integer, intent(in) :: arg
     print *, arg
   end subroutine

   subroutine foo_c(arg)
     complex, intent(in) :: arg
     print *, arg
   end subroutine
end module foo_mod
-------------------------
```

and a second file

```
--- file  f04.f90 ---------
program f04
  use foo_mod
  implicit none

  complex, parameter :: x= CMPLX(0,2.0*ACOS(0.0))
  integer, parameter :: y=5

  CALL foo(x)
  CALL foo(y)

end program f04
-----------------------

$./a.out
 (  0.0000000    ,  3.1415927    )
           5
```

2.5 f05.f90

f05.f90

```
--------- file f05.f90 -----------------------
program f05
!-- Overload 2 subroutine names as above but using one file
!-- but need to use std=gnu to compile

  implicit none

  complex, parameter :: x= CMPLX(0,2.0*ACOS(0.0))
  integer, parameter :: y=5
```

```
  interface foo
     procedure foo_c, foo_i
  end interface foo

  CALL foo(x)
  CALL foo(y)

contains
  subroutine foo_i(arg)
    integer, intent(in) :: arg
    print *, arg
  end subroutine

  subroutine foo_c(arg)
    complex, intent(in) :: arg
    print *, arg
  end subroutine

end program f05
```

2.6 f06.f90

f06.f90

```
--------- file f06.f90 -----------------------
!-- It is possible to put the module inside
!-- the same file as the program that uses it
!-- if it is only the program that needs it
program f06
  use foo_mod
  implicit none

  complex, parameter :: x= CMPLX(0,2.0*ACOS(0.0))
  integer, parameter :: y=5

  CALL foo(x)
  CALL foo(y)

end program f06

module foo_mod
  implicit none
  interface foo
    procedure foo_i,foo_c
  end interface
contains
   subroutine foo_i(arg)
     integer, intent(in) :: arg
     print *, arg
   end subroutine

   subroutine foo_c(arg)
     complex, intent(in) :: arg
     print *, arg
   end subroutine
end module foo_mod
--------- end file f06.f90 -----------------------
```

```
$./a.out
 (  0.0000000    ,  3.1415927    )
          5
```

2.7 f07.f90

f07.f90

```
--------- file f07.f90 -----------------------
!-- using select_int_kind
program f07
  implicit none

  integer(selected_int_kind(5))  :: i=16
  integer(selected_int_kind(10)) :: j=16

  CALL foo(i)
  CALL foo(j)

contains
  subroutine foo(arg)
    integer(selected_int_kind(5)), intent(in) :: arg
    print *, arg
  end subroutine foo

end program f07
--------- end file f07.f90 -----------------------
```

2.8 f08.f90

f08.f90

```
!-- showing how to use Fortran for vectored operations
!-- equations work on vectors, no need for loop
!-- showing how to use Fortran for vectored operations
!-- equations work on vectors, no need for loop
program f08
  implicit none

  integer, parameter :: N = 7
  real   , parameter :: D(N) = [-0.2,1.0,1.5,3.0,-1.0,4.2,3.1]
  real   , parameter :: H(N) = [2.1,2.4,1.8,2.6,2.6,2.2,1.8]
  real   , parameter :: pi = 2.0* ACOS(0.0)
  real               :: V(N)

  V = (1.0/12.0)*pi*(D**2)*H
  print *, v


end program f08

$./a.out
2.19911486E-02 0.62831855 1.0602875  6.1261053  0.68067843  10.159910  4.5286055
```

2.9 f08a.f90

f08_a.f90

```
!-- As above, but uses allocatable
!-- showing how to use Fortran for vectored operations
!-- equations work on vectors, no need for loop
program f08_a
  implicit none

  integer, parameter :: N = 7
  real, parameter :: D(N) = [-0.2,1.0,1.5,3.0,-1.0,4.2,3.1]
  real, parameter  :: H(N) = [2.1,2.4,1.8,2.6,2.6,2.2,1.8]
  real, parameter :: pi = 2.0* ACOS(0.0)
  real, allocatable :: V(:)

  V = (1.0/12.0)*pi*(D**2)*H
  print *, v

end program f08_a
```

2.10 f09.f90

f09.f90

```
!-- print exp(n*I*Pi/4) for n=1..10, where I is the sqrt(-1)
program f09
   implicit none
   INTEGER, PARAMETER :: DP = KIND(0.0D0)
   integer :: n
   real(kind=DP), parameter  :: pi = 4.D0*DATAN(1.D0)
   complex, parameter        :: I = CMPLX(0, 1)
   DO n = 1,10
      print *, exp(n*I*pi/4.0D0)
   END DO
end program f09

$gfortran -std=f2003 -Wextra -Wall -pedantic -fcheck=all \
         -march=native  -Wsurprising -Wconversion  f09.f90
$./a.out
 ( 0.70710678118654757     , 0.70710678118654746     )
 ( 6.12303176911188629E-017,  1.0000000000000000     )
 (-0.70710678118654746     , 0.70710678118654757     )
 ( -1.0000000000000000     , 1.22460635382237726E-016)
 (-0.70710678118654768     ,-0.70710678118654746     )
 (-1.83690953073356589E-016, -1.0000000000000000     )
 ( 0.70710678118654735     ,-0.70710678118654768     )
 (  1.0000000000000000     ,-2.44921270764475452E-016)
 ( 0.70710678118654768     , 0.70710678118654735     )
 ( 3.06151588455594315E-016,  1.0000000000000000     )
$
```

2.11 f10.f90

f10.f90

```
!-- shows how to declare 2D array in place
!-- thanks to help from James Van Buskirk and glen herrmannsfeldt
!-- from comp.lang.fortran
program f10
implicit none

INTEGER, dimension(2,3)::A = &
   reshape(   &
     [2,3,4 , &
      4,0,7], &
      shape(A) , order = [2,1])

integer :: i

do i=LBOUND(A, 1),UBOUND(A, 1)
   print *,A(i,:)
end do

end program f10


>./a.out
           2           3           4
           4           0           7
```

2.12 f11.f90

f11.f90

Example of using `advance='no'`

```
PROGRAM f11
   IMPLICIT none
   INTEGER :: i
   INTEGER, DIMENSION(2,3)::B = &
        RESHAPE([2,3,4 , &
                 4,0,7], &
        SHAPE(B) , ORDER = [2,1])

  DO i=1,2
     WRITE(*,'(A)',ADVANCE='no') '['
     WRITE(*,  '(*(I2))', ADVANCE='no')  B(i,:)
     WRITE(*,'(A)' ) ' ]'
  END DO
END PROGRAM f11

>./f11
[ 2 3 4 ]
[ 4 0 7 ]
```

3 build Fortran binding to gsl

```
trying to build gsl fortran, but get errors

$./configure --f90 gfortran --gsl /usr/local
  Using /usr/bin/gfortran as Fortran compiler
  Using /usr/bin/gcc as C compiler
  Using GSL library located in /usr/local/lib
  Installation target path is /usr/local.
Configuration successful. Now run make to build,
or make test to run test suite. Enjoy!

$make
gfortran   -c  fgsl.f90
gcc -c -I/usr/local/include   -o fgsl_utils.o fgsl_utils.c
fgsl_utils.c:9:28: fatal error: gsl/gsl_odeiv2.h: No such file or directory
compilation terminated.
make: *** [fgsl_utils.o] Error 1
$
```

4 compiling with pgplot

installed pgplot using synaptic package manager. This is below how to build the fortran example shown at the pgplot website

```
----- ex1.f --------------------
PROGRAM EX1
      INTEGER PGOPEN, I
      REAL XS(9), YS(9), XR(101), YR(101)
C Compute numbers to be plotted.

      DO 10 I=1,101
          XR(I) = 0.1*(I-1)
          YR(I) = XR(I)**2*EXP(-XR(I))
 10   CONTINUE
      DO 20 I=1,9
          XS(I) = I
          YS(I) = XS(I)**2*EXP(-XS(I))
 20   CONTINUE

C Open graphics device.
      IF (PGOPEN('?') .LT. 1) STOP
C Define coordinate range of graph (0 < x < 10, 0 < y < 0.65),
C and draw axes.
      CALL PGENV(0., 10., 0., 0.65,  0,  0)
C Label the axes (note use of \u and \d for raising exponent).
      CALL PGLAB('x', 'y', 'PGPLOT Graph: y = x\u2\dexp(-x)')
C Plot the line graph.
      CALL PGLINE(101, XR, YR)
C Plot symbols at selected points.
      CALL PGPT(9, XS, YS, 18)
C Close the graphics device.
```

```
      CALL PGCLOS
      END
```

The command used is

```
gfortran ex1.f -L/usr/lib -L/usr/lib/i386-linux-gnu -lpgplot -lcpgplot -lX11 -lpng
\end{Verbatim]

This was done on
\begin{Verbatim}[samepage=true]
$uname -a
Linux me-VirtualBox 3.0.0-12-generic #20-Ubuntu SMP Fri Oct 7 14:50:42 UTC 2011 i686 i686 i386 GNU/Linux

$gfortran -v
Using built-in specs.
COLLECT_GCC=gfortran
COLLECT_LTO_WRAPPER=/usr/lib/gcc/i686-linux-gnu/4.6.1/lto-wrapper
Target: i686-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.6.1-9ubuntu3' --with-bugurl=file:///u
Thread model: posix
gcc version 4.6.1 (Ubuntu/Linaro 4.6.1-9ubuntu3)
$
```

5 install gfortran

```
>which gfortran
>gfortran
The program 'gfortran' is currently not installed. You can install it by typing:
sudo apt-get install gfortran
>sudo apt-get install gfortran
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  cpp-4.8 gcc-4.8 gcc-4.8-base gfortran-4.8 libasan0 libatomic1 libc-dev-bin libc6-dev libgcc-4.8-dev libgcc1
  libgfortran-4.8-dev libgfortran3 libgomp1 libitm1 libquadmath0 libstdc++6
Suggested packages:
  gcc-4.8-locales gcc-4.8-multilib libmudflap0-4.8-dev gcc-4.8-doc libgcc1-dbg libgomp1-dbg libitm1-dbg
  libatomic1-dbg libasan0-dbg libtsan0-dbg libbacktrace1-dbg libquadmath0-dbg libmudflap0-dbg gfortran-multilib
  gfortran-doc gfortran-4.8-multilib gfortran-4.8-doc libgfortran3-dbg glibc-doc
The following NEW packages will be installed:
  gfortran gfortran-4.8 libc-dev-bin libc6-dev libgfortran-4.8-dev
The following packages will be upgraded:
  cpp-4.8 gcc-4.8 gcc-4.8-base libasan0 libatomic1 libgcc-4.8-dev libgcc1 libgfortran3 libgomp1 libitm1
  libquadmath0 libstdc++6
12 upgraded, 5 newly installed, 0 to remove and 193 not upgraded.
Need to get 28.0 MB of archives.
After this operation, 38.0 MB of additional disk space will be used.
Do you want to continue [Y/n]?
Get:1 http://archive.ubuntu.com/ubuntu/ saucy-updates/main libquadmath0 i386 4.8.1-10ubuntu9 [218 kB]
Get:2 http://archive.ubuntu.com/ubuntu/ saucy-updates/main libitm1 i386 4.8.1-10ubuntu9 [36.8 kB]
Get:3 http://archive.ubuntu.com/ubuntu/ saucy-updates/main libgomp1 i386 4.8.1-10ubuntu9 [29.0 kB]
Get:4 http://archive.ubuntu.com/ubuntu/ saucy-updates/main gcc-4.8-base i386 4.8.1-10ubuntu9 [16.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu/ saucy-updates/main libstdc++6 i386 4.8.1-10ubuntu9 [335 kB]
Get:6 http://archive.ubuntu.com/ubuntu/ saucy-updates/main libgfortran3 i386 4.8.1-10ubuntu9 [324 kB]
Get:7 http://archive.ubuntu.com/ubuntu/ saucy-updates/main libatomic1 i386 4.8.1-10ubuntu9 [9,694 B]
Get:8 http://archive.ubuntu.com/ubuntu/ saucy-updates/main libasan0 i386 4.8.1-10ubuntu9 [76.2 kB]
Get:9 http://archive.ubuntu.com/ubuntu/ saucy-updates/main cpp-4.8 i386 4.8.1-10ubuntu9 [5,657 kB]
Get:10 http://archive.ubuntu.com/ubuntu/ saucy-updates/main gcc-4.8 i386 4.8.1-10ubuntu9 [6,450 kB]
```

```
Get:11 http://archive.ubuntu.com/ubuntu/ saucy-updates/main libgcc-4.8-dev i386 4.8.1-10ubuntu9 [2,795 kB]
Get:12 http://archive.ubuntu.com/ubuntu/ saucy-updates/main libgcc1 i386 1:4.8.1-10ubuntu9 [53.7 kB]
Get:13 http://archive.ubuntu.com/ubuntu/ saucy-updates/main libgfortran-4.8-dev i386 4.8.1-10ubuntu9 [390 kB]
Get:14 http://archive.ubuntu.com/ubuntu/ saucy/main libc-dev-bin i386 2.17-93ubuntu4 [74.9 kB]
Get:15 http://archive.ubuntu.com/ubuntu/ saucy/main libc6-dev i386 2.17-93ubuntu4 [5,519 kB]
Get:16 http://archive.ubuntu.com/ubuntu/ saucy-updates/main gfortran-4.8 i386 4.8.1-10ubuntu9 [5,984 kB]
Get:17 http://archive.ubuntu.com/ubuntu/ saucy/main gfortran i386 4:4.8.1-2ubuntu3 [1,206 B]
Fetched 28.0 MB in 20s (1,391 kB/s)
(Reading database ... 147551 files and directories currently installed.)
Preparing to replace libquadmath0:i386 4.8.1-10ubuntu8 (using .../libquadmath0_4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement libquadmath0:i386 ...
Preparing to replace libitm1:i386 4.8.1-10ubuntu8 (using .../libitm1_4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement libitm1:i386 ...
Preparing to replace libgomp1:i386 4.8.1-10ubuntu8 (using .../libgomp1_4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement libgomp1:i386 ...
Preparing to replace gcc-4.8-base:i386 4.8.1-10ubuntu8 (using .../gcc-4.8-base_4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement gcc-4.8-base:i386 ...
Setting up gcc-4.8-base:i386 (4.8.1-10ubuntu9) ...
(Reading database ... 147551 files and directories currently installed.)
Preparing to replace libstdc++6:i386 4.8.1-10ubuntu8 (using .../libstdc++6_4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement libstdc++6:i386 ...
Preparing to replace libgcc1:i386 1:4.8.1-10ubuntu8 (using .../libgcc1_1%3a4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement libgcc1:i386 ...
Setting up libgcc1:i386 (1:4.8.1-10ubuntu9) ...
Setting up libstdc++6:i386 (4.8.1-10ubuntu9) ...
Processing triggers for libc-bin ...
(Reading database ... 147551 files and directories currently installed.)
Preparing to replace libgfortran3:i386 4.8.1-10ubuntu8 (using .../libgfortran3_4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement libgfortran3:i386 ...
Preparing to replace libatomic1:i386 4.8.1-10ubuntu8 (using .../libatomic1_4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement libatomic1:i386 ...
Preparing to replace libasan0:i386 4.8.1-10ubuntu8 (using .../libasan0_4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement libasan0:i386 ...
Preparing to replace cpp-4.8 4.8.1-10ubuntu8 (using .../cpp-4.8_4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement cpp-4.8 ...
Preparing to replace gcc-4.8 4.8.1-10ubuntu8 (using .../gcc-4.8_4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement gcc-4.8 ...
Preparing to replace libgcc-4.8-dev:i386 4.8.1-10ubuntu8 (using .../libgcc-4.8-dev_4.8.1-10ubuntu9_i386.deb) ...
Unpacking replacement libgcc-4.8-dev:i386 ...
Selecting previously unselected package libgfortran-4.8-dev:i386.
Unpacking libgfortran-4.8-dev:i386 (from .../libgfortran-4.8-dev_4.8.1-10ubuntu9_i386.deb) ...
Selecting previously unselected package libc-dev-bin.
Unpacking libc-dev-bin (from .../libc-dev-bin_2.17-93ubuntu4_i386.deb) ...
Selecting previously unselected package libc6-dev:i386.
Unpacking libc6-dev:i386 (from .../libc6-dev_2.17-93ubuntu4_i386.deb) ...
Selecting previously unselected package gfortran-4.8.
Unpacking gfortran-4.8 (from .../gfortran-4.8_4.8.1-10ubuntu9_i386.deb) ...
Selecting previously unselected package gfortran.
Unpacking gfortran (from .../gfortran_4%3a4.8.1-2ubuntu3_i386.deb) ...
Processing triggers for man-db ...
Setting up libquadmath0:i386 (4.8.1-10ubuntu9) ...
Setting up libitm1:i386 (4.8.1-10ubuntu9) ...
Setting up libgomp1:i386 (4.8.1-10ubuntu9) ...
Setting up libgfortran3:i386 (4.8.1-10ubuntu9) ...
Setting up libatomic1:i386 (4.8.1-10ubuntu9) ...
Setting up libasan0:i386 (4.8.1-10ubuntu9) ...
Setting up cpp-4.8 (4.8.1-10ubuntu9) ...
Setting up libgcc-4.8-dev:i386 (4.8.1-10ubuntu9) ...
Setting up gcc-4.8 (4.8.1-10ubuntu9) ...
Setting up libgfortran-4.8-dev:i386 (4.8.1-10ubuntu9) ...
Setting up libc-dev-bin (2.17-93ubuntu4) ...
Setting up libc6-dev:i386 (2.17-93ubuntu4) ...
Setting up gfortran-4.8 (4.8.1-10ubuntu9) ...
Setting up gfortran (4:4.8.1-2ubuntu3) ...
```

```
update-alternatives: using /usr/bin/gfortran to provide /usr/bin/f95 (f95) in auto mode
Processing triggers for libc-bin ...
>
```

now

```
>which gfortran
/usr/bin/gfortran
>gfortran -v
Using built-in specs.
COLLECT_GCC=gfortran
COLLECT_LTO_WRAPPER=/usr/lib/gcc/i686-linux-gnu/4.8/lto-wrapper
Target: i686-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.8.1-10ubuntu9' --with-bugurl=file:///
Thread model: posix
gcc version 4.8.1 (Ubuntu/Linaro 4.8.1-10ubuntu9)
```

6 my Fortran cheat sheet notes

1. use this to compile

   ```
   gfortran   -fcheck=all -Wall -Wconversion -Wextra -Wconversion-extra -pedantic
   ```

2. ```
   ldd -- shows what libraries a program linked to
   nm  -- shows what symbols in the shared library of object file and type of symbol

   reference for the kind  data
   http://docs.oracle.com/cd/E19059-01/stud.10/819-0492/4_f95.html

   compile options to use
   gfortran -Wextra -Wall -pedantic -fcheck=all -fcheck=do -fwhole-file
   -fcheck=pointer-O2 -funroll-loops -ftree-vectorize  -Wsurprising
   -Wconversion

   I am getting
   >gcc -I/usr/include -c glut_fonts.c
   In file included from /usr/include/features.h:389:0,
                    from /usr/include/inttypes.h:26,
                    from /usr/include/GL/glext.h:5386,
                    from /usr/include/GL/gl.h:2085,
                    from /usr/include/GL/freeglut_std.h:120,
                    from /usr/include/GL/glut.h:17,
                    from glut_fonts.c:1:
   /usr/include/gnu/stubs.h:7:27: fatal error: gnu/stubs-32.h: No such file or directory
   compilation terminated.

   I do not know how this happend, but /usr/include/gnu/stubs-32.h was missing.
   So I copied one from different distributions and now it works.


   To find which file comes from which dep package see
   http://www.debian.org/distrib/packages#search_contents

   to find which file comes from which RPM package see
   ```
   ```

```
static link note
------------------
2.7 Influencing the linking step

These options come into play when the compiler links object
files into an executable output file. They are meaningless if
the compiler is not doing a link step.

-static-libgfortran
    On systems that provide libgfortran as a shared and a static
    library, this option forces the use of the static version. If no
    shared version of libgfortran was built when the compiler was
    configured, this option has no effect.

getting link error when doing static
missing __gxx_personality_v0
```

3. to have a loop exit loop type construct

```
      my_loop: do
         read (7, '(A)') V
         if (V(1) /= "#") exit my_loop
      end do ReadComments
      backspace (7)
```