**Free Fortran Compilers**

There are a number of free Fortran 77 and 90 compilers available on the [net](http://sunsite.informatik.rwth-aachen.de/fortran/append-f.html). The one I have been using in my Fortran courses at York is **GNU**, which implements Fortran 77 and adds several Fortran 90 features. Thanks to Prof. Clive Page (Dept of Physics & Astronomy, University of Leicester, UK) for providing the compiler and for valuable advice on Fortran in general.

You can download the 1999 version of this compiler (version 2.95 of gcc) along with the **SLATEC** library (Version 4.1, July 1993), from this page. The package runs under all (32-bit) versions of Windows, including XP. All the needed files are packed in one zipped file (Fort99.zip) of about 6MB.

(If for some reason you need the older DOS/EMX version, which does not include a library and does not run under Windows XP, then you can download it from my [old](http://www.cse.yorku.ca/%7Eroumani/fortran/ftnOld.htm) page.)

**DOWNLOAD**

* Create the directory \F  
  The new folder must be immediately under the root of your hard disk. You can do this by double-clicking *MyComputer*, then double-clicking your hard drive (usually C:), and then selecting *New Folder* from the *File* menu and calling the folder F.
* Download the file [Fort99.zip](http://www.cse.yorku.ca/%7Eroumani/fortran/gnu99/Fort99.zip) (5,820,239 bytes).  
  You can do this by right-clicking the mouse on the above link, and choosing *Save Target As...*. In the *Save As* window that appears, locate the F folder, and save the file in it.
* Unzip the downloaded file into \F.  
  Yon can do this by locating the file (starting from *MyComputer*) and simply double-clicking it to launch the zip/unzip program. Specify that all files should be extracted to the F folder with their path names preserved (this is usually the default). If you don't have a zip/unzip program, download one from the Internet.
* Set LIBRARY\_PATH to \F\G77\lib and include \F\G77\bin in your path. How these two environment variables are set varies from one version of Windows to another:
  + For the **Windows 95/98/Me** line: These variables are set in the autoexec.bat file, which is stored in the *root* directory of your hard disk; i.e. \. Hence, you need to edit this file and add to the two lines:
  + PATH=\F\G77\bin;%PATH%

SET LIBRARY\_PATH=\F\G77\lib

You can do this by opening this file in Notepad or any other editor, adding the two lines at the end, and then saving the changes.

* + For the **Windows NT/2000/XP** line: These variables can be set by going to the Control Panel, selecting System, and then locating the environment (or advanced, environment) section. You need to add a new variable with name LIBRARY\_PATH and set its value to \F\G77\lib. Similarly add a new variable named PATH and set its value to \F\G77\bin or append ;\F\G77\bin to its value if it already exists.

Finally, restart your computer to allow the environment changes to take effect.

**USAGE**

You store your programs in the \F\York directory, compile them using: f2exe, and create library object files using f2lib. Here is a very short program to test the compiler and the configuration:

program Convert

implicit none

! -----------------------------------------------Declare

real\*4 tempC, tempF, FACTOR

integer\*2 ZERO\_SHIFT

parameter (ZERO\_SHIFT = 32, FACTOR = 5./9.)

! -----------------------------------------------Input

print\*, "Enter the temperature in Fahrenheit ..."

read\*, tempF

! -----------------------------------------------Compute

tempC = FACTOR \* (tempF - ZERO\_SHIFT)

! -----------------------------------------------Output

print\*, "The corresponding Centigrade temperature is "

print\*, tempC, " degrees."

end

Use any editor to create this program (simply copy and paste) and save it as a text file in the \F\York directory under the name test.for. You can, of course, use any editor you like as long as you can save the file in text format and with the extension you want. Notepad, for example, uses text but insists on using the txt extension (unless you override by double-quoting) while MS-Word insists on its propriety format (unless you explicitly override). I highly recommend using the Crimson editor, which can be downloaded from the on-line **Lab-1** (see below).

Compile and run your program from the DOS command prompt by typing:

cd \F\York

f2exe test

test

If the first command returned an error then the directory was not created (or named) correctly. If the second command was not recognized, or complained that a library is missing, then the environment variables were not set correctly (you can issue the set command to inspect all environment variables).

More information on using the compiler can be found in the on-line **Labs** at the [*Fortran@York*](http://www.cse.yorku.ca/%7Eroumani/fortran/index.html) site.

**LANGUAGE**

The \F\G77\doc directory has a detailed reference to the language, which is largly [ANSI Fortran-77](http://www.fortran.com/fortran/F77_std/rjcnf0001.html) but with some Fortran-90 features added (see below).

The above *Fortran@York* site contains a quick reference guide, lab, and SLATEC usage examples. If you are already familiar with Fortran then the following points may be all you need to know about this compiler:

1. **Control Structures**  
   You can use either the old (goto-based) or the new (structured) control flow (or mix them in the same program). Support of the "ugly goto" is meant for existing code only, and any new development should avoid it.
2. **Style**  
   You can write your source using either the old style code (column 7) or the newer free-form.
3. **Compilation Command**  
   The above f2exe command is just a batch file that invokes g77, the "real" compilation command. The command:

**g77 -ffree-form prog.for -oprog.exe**

directs the compiler to compile the file prog.for and stores the output in the file prog.exe. The -ffree-form switch indicates free-form style (remove it if you are using the old style).

1. **Comments**  
   In free-form style, use ! for both full-line and in-line comments. In the old style, use a "C" in column-1.
2. **Statement Continuation**  
   In free-form style, you can continue a statement on the next line by ending it with an ampersand "&". In the old style, put a character in column-6.
3. **Path Separator**  
   When referring to files (e.g. in the file=' ' parameter of the OPEN statement) use a forward slash "/" or two consecutive backslashes "\\" rather than a backslash to delimit directories. This is because the backslash "\" denotes an escape sequence in strings.
4. **I/O Unit Numbers**  
   Not all unit numbers are allowed in the OPEN statement. In particular, unit 5 is "pre-connected" to the keyboard. Units 10 through 99 seem to work well with disk files.
5. **Fortran-90 Features**  
   These include: Automatic arrays in subprograms, zero length strings, character constants may be enclosed in double quotes (") as well as single quotes, cycle and exit, the DOUBLE COMPLEX type, DO WHILE, the END decoration, KIND, IMPLICIT NONE, INCLUDE statements, list-directed and namelist I/O on internal files, binary, octal, and hex constants, `O' and `Z' edit descriptors, NAMELIST, OPEN specifiers STATUS='REPLACE', the FILE= specifier may be omitted in an OPEN statement if STATUS='SCRATCH' is supplied, relational operators <, <=, ==, /=, > and >= may be used instead of .LT., .LE., .EQ., .NE., .GT. and .GE. respectively, SELECT CASE (but not for character types).
6. **Separate Compilation of Subprograms**  
   Your main program is recognized by the program statement, as in the Convert program above. The subprograms (functions and subroutines) can be included in the same file as the main program (in which case you can compile everything in one shot) or can be stored in separate file(s). It is recommended that you store general reusable subprograms in separate files so that you can reuse them (without recompiling them) in future projects. To compile a file that contains only subprograms (no program statement), use the f2lib command, which generates object files, one per sub, in the mine directory, e.g.

**f2lib util**

will compile (without linking) the subprogram in util.for and store the output (an object file) in the file util.o. f2lib is just a batch file that invokes the g77 command with the **-c** (compile-only) switch, viz.

**g77 -c -ffree-form util.for -o..\mine\util.o**

A program that uses pre-compiled object files can be compiled (and linked to them) by simply referring to them in the compilation command:

**g77 -ffree-form prog.for -oprog.exe ..\mine\\*.o**

The above command searches all object files in mine to resolve any missing reference in prog.for.

1. **Separate Compilation of Subprograms, automated**  
   The supplied f2exe and f2lib batch files take care of separate compilation and delayed linking with object files and with the SLATEC subprograms. You don't have to directly issue the g77 command unless you use the old columnar style or you want to change one of the switches or directories.
2. **Assembly Listing**  
   The **-S** (capital S) switch allows you to see a listing of the generated assembly code.