Lecture notes: Compiling and linking on Pets

Tom Aage Jelmert
TPG 5100 Applied data

Log on to the computer

In this course we use the "Pets" machine.

Log on to the computer by:

pets.ivt.ntnu.no

Use program: "SSH secure" or "XWin32". For details: www.ipt.ntnu.no/~info. The use of Xwin32 is recommended since it has better graphical capabilities.

Operating system

The operating system is LINUX. It is case-sensitive, which means upper and lower case letters are considered different characters. Thus the file Out.dat is different from out.dat.

Extension for Fortran data files are not necessary. For clarity we recommend the use of either .d or .dat. The extension will highlight that this is a data file. A Fortran data file is a collection of data that can be used by a Fortran program (input file) or the result of a Fortran run (output file).

The extension for a Fortran program file is .f , f90 or .f95, where f denotes Fortran program file and 90 or 95 highlights that we use Fortran90 or Fortran95 standard.

Example:

Log on to the computer:

Make a directory for this course by punching: mkdir "DirectoryName"

Example:

mkdir AppliedData (Make directory)

Check the existence of "AppliedData"

punch: ls or dir (list the content of the directory)

Enter your new directory:

cd AppliedData (change directory to)

Check that you are in the right directory:

punch: pwd (pwd means present working directory)

Text-editor
Start the editor by: emacs -nw "filename" & if you log in by SSh. The option –nw is not necessary if you log in by XWIN32.

The sign "&" give the instruction that the text editor shall be in the background.

Example: Make an input file called inp.dat and save the integer 7 and the real number 8.5.

```
emacs input.dat &
    8.5  7
```

Save the data: CTRL-x CTRL-s in emacs.

Leave emacs: CTRL –x CTRL -c

Confirm the existence of input.dat:

ls or dir (ls means list content of the present directory)

Check the existence of the data saved:

emacs input.dat &

Computer language

The majority of exercises will use the FORTRAN language. We also discuss VBA (Visual Basic Applications) and Matlab. The latter is available both on UNIX and Windows. VBA runs on Windows only.

Compile and link programs:

Compiler: A computer program that converts a program unit written in a computer language such as Fortran (symbolic element) into machine code (object code).

Link: The process of combining object modules from program units and computer libraries to form an executable program (executable element). One may also link the program to a prewritten program library. In this course we will use the NAG library.

Procedure:

Many competing Fortran compilers are available.

The Fortran compilers available on Pets have been called: "ifort" f or "f95". The latter has the advantage that is the standard Linux compiler. It is free and can be downloaded to any machine with a Linux operating system. In my experience the debugging system is slightly better. It does not enjoy the speed of "ifort" which has been designed for parallelization.

Suppose there exists a Fortran program called: "MyFirstProg.f90" (extension ".f90" denotes Fortran program according to the Fortran90 standard). This is the main program.

In addition, suppose the main program MyFirstProg.f90 calls another program which has been called MyFirstSub.f90. This is a subprogram.

Suppose both programs are necessary and sufficient to complete the task the program has been designed to solve.

Note: The Fortran compiler is case insensitive. Upper and lower case letters are treated the same way. Thus the variable "Permeability" is the same as "permeabiliTY".

Step 1: Compile the program units

Command:

ifort[1] -c MyFirstProg.f90, MyFirstSub.f90

or sequentially

ifort -c MyFirstProg.f90

and

ifort - c MyFirstSub.f90

The minus sign denotes an option for the compiler. Option " –c" denotes compilation. (object-files are produced). The object-files exist as separate units. They are neither linked together nor to the computer libraries and hardware.

The names are arbitrary. We recommend names that describe the use of the program unit.

[1] "Ifort" has been used as an example. The alternative is to replace it by "f95"

Result: Converts the program units into machine code by use of the Fortran90 standard.

MyFirstProg.o
MyFirstSub.o

Extension ".o" denotes object file. These are in machine code but cannot be executed. The c-option is handy for writing and initial debugging of a symbolic element. The c-option is mandatory if you want to build a library of program units that may be shared between several programs. The library objectiv files can later be linked to any program when needed.

Step 2: Compile and Link the resulting object-files.

When all obvious Fortran errors have been removed, then try to make an executable element. Combine the program units into a complete program:

Command:

ifort[2] -o Program MyFirstProg.f90 MyFirstSub.90

Option "–o" denotes output (which is the executable element). The command must include the name of the executable element and a list of the object files to be included. The name of the program could be anything. In this case the name is "Program". This name is not recommended. It is better practice to use a name that describes the purpose of the program. Examples could be: CoreAnalysis, Stacking, WellTest etc. It may also be a good idea to give the program the same name as the main program.

Result: The program units are combined to make a complete program that may be executed.

Program

No extension is associated with the name of the program.

Many compilers will generate program with the extension .exe. Some also use the extension .com

Extension ".exe" denotes executable element.

Step 3: Run the program (executable element).

Punch: Program (The name of the program)

---

[2] Note: Object files generated by i"fort" cannot be linked to object files obtained by "f95" or the the other way around.

Step 4: Bebug the program

This phase of the program development is the most time consuming. Logical errors are hard to detect. Check the program by giving it a problem with a known solution. If the program fails to produce the required solution, give the program a small problem and check the calculation of selected variables by use of EXCEL or a calculator.