PHYS-4007/5007: Computational Physics

Fortran Programming in Linux
Help with Linux and Fortran on the Brown Hall 264 Computers

1 Getting Started with Linux.

The information of logging in on the Linux side of the computers in Brown Hall 264 can be found on your **Linux Computer Account Information** sheet passed out in class. As explained on that sheet, one should have made two directories in your login directory using

> mkdir fortran

> mkdir tex

Note that on the lines above the '>' represents the Linux (referred to as Unix from this point forward) prompt — **do not type this in!** ('>' is used here since each person's prompt is based upon the current directory one is in.) Now do a directory listing using

> ls

this gives a directory listing of the current directory (*i.e.*, "folder" in the Microsoft world). Now enter the following four command in succession

> ls –px

> ls –pxa

> ls –l

> ls –la

The letters after the '–' are called 'flags' in Unix: '–p' appends characters to the file/directory names ('/' for directory, '—' for link, etc.); '–x' lists the entries by lines instead of columns; '–l' shows the long listing of the files which shows what type of file it is, the protections set for the file, it's size in bytes, etc.; and '–a' shows hidden files (those whose file name starts with a dot ('.') as well as normal files). Note that one can include more than one 'flag' keyword after the '–' as shown above. One can get detailed information about any Unix command using the 'man' (for manual) command:

> man ls

See a list of other common Unix commands in Subsection D of Appendix A of the course notes.

2 Creating, Compiling and Running Fortran Code

It's a good idea to keep your Unix files organized so they are easily found when you need them. This is why I had you created the fortran and tex directories above. At this point cd (for change directory) into your fortran subdirectory (directories below your login directory are called 'subdirectories'):

> cd fortran

Let's now create a Fortran file using emacs:

> emacs mycode.f &

note that the '&' sign put this job in background so that you can still access Linux in the terminal window. There are one of two ways we can input the text presented below. First, we could just go to the Tutorial web page linked to the course web page and click on the first 'mycode.f' link; then select, copy and paste into the file opened in your emacs session (make sure that the Fortran command lines start in column 7 in the file). Or we can just type the lines presented below into the emacs GUI (Graphic User Interface) window (*note:* do not include the first line of the sequence of numbers):

```
123456789T123456...
      PROGRAM MYCODE
C
C This is my first Fortran code.
C
      INTEGER I, J, I_INDEX(100)
      REAL PI, LLCOEFF(100)
      REAL*8  DPI, SSCOEFF(100), BBOUT(100)
C
      PI = 3.14159
      DPI = 3.14159265359D0
C
C Calculate integer and single precision arrays.
C
      DO 52 I = 1, 100
         I_INDEX(I) = I*2
         LLCOEFF(I) = FLOAT(I**2) * SIN(PI*FLOAT(I)/10.)
  52  CONTINUE
C
C Calculate double precision arrays.  This DO loop has an error.
C
```

```
      DO 1005 J = 1, 100
         SSCOEFF(J) = COS(DPI*DFLOAT(J)/8.0D0)
         BBOUT(J) = SSCOEFF(J) * DEXP(-DFLOAT(J)/300.D0)
  52  CONTINUE
C
C Output this data to the screen in ordered columns.
C
C This next DO loop has an error in it.
C
      DO 2344 I = 1, 200
         WRITE(*, 2340) I, I_INDEX(I), LLCOEFF(I), SSCOEFF(I),
     1      BBOUT(I)
C
C This FORMAT statement has two errors in it,
C one minor and one severe.
C
 2340    FORMAT(2X,2I5,2X,F8.4,2X,1PE12.2)
 2344 CONTINUE
C
      STOP
      END
```

In the code above note that I purposely included a few programming errors, some you will see when compiling and others when you run the code. You will need to fix these in order for the code to run successfully (see the next section). Also note that one should be careful with variable types when carrying out calculations:

- One should only have integer-type for integer statement arithmetic,
- only floats (*i.e.*, REAL) for single-precision calculation statements,
- only doubles (*i.e.*, REAL*8 or DOUBLE PRECISION) for double-precision calculation statements,
- only characters in string statements, etc.

On the Tutorial web page linked to the course web page, there is a PDF document that lists some of the standard math, type conversion, and operational functions found in Fortran. Print this out and keep it as reference for when you are doing Fortran programming. Make sure you pay attention to these tables when doing your homework, exams, and project for this course. Information on working with character strings in Fortran can be found in Appendix B of the course notes.

In your emacs editor GUI, either press the 'floppy disk' icon near the top of the GUI or select the 'File' icon pull-down menu near the top left of the GUI and select save. Once done, select the 'Exit Emacs' item in the 'File' pull-down menu to close the emacs GUI.

At the Unix prompt from your fortran subdirectory, enter the following command:

> gfortran –o mycode.exe mycode.f

Assuming there are no compile errors, this command will make an executable file called mycode.exe. To run this code one simply types at the Unix prompt (remember this is labeled here as '>'):

> ./mycode.exe

Here the './' tells Unix that the executable file is in the current directory. This will result in an output of 200 lines of numbers in column format showing the results of the calculations in your code. Some of the output will look 'strange' due to the errors I purposely included in this code. We will address these errors below.

3 Fixing Our Sample Fortran Code

You should have noted the following errors, either while compiling or during the operation of the code. I list these errors now:

- The 'DO 1005' loop has an error in the numbered CONTINUE statement that closes this DO loop. As can be seen, instead of '52 CONTINUE' we should have written '1005 CONTINUE' — this error occurred since I did a clip and paste from the previous DO loop and I forgot to change the label number associated with CONTINUE.
- In the 'DO 2344' loop, variable I steps from 1 (one) to 200, yet the variables that are being printed are only 100 element arrays. As such, there is no telling what would be output for the I = 101 to I = 200 elements in this output — it could be zeros, or it could be "jibberish."
- In the '2340 FORMAT' statement, our first 2 variables are integers and we allowed 5 spaces for each integer which is more than sufficient for the values stored in these variables. However, the first floating point number array, LLCOEFF, will contain values that will be too large for the 8 spaces we have allowed with the 4 spaces after the decimal point. This results in '*******' being printed instead of numbers for some of the values. We can fix this by changing the '8' to a '10' in the 'F' FORMAT code (*i.e.*, F10.4).

- There are two arrays that are being printed out in scientific format, SSCOEFF(I) and BBOUT(I), yet the 2340 FORMAT statement has only one format code listed for these variables (*i.e.*, 1PE12.2). As such, this will either cause the code to crash during a run or produce strange output for the subsequent numbers in the following lines of output. To fix this, we just need to change 'textsf1PE12.2' to 'textsf1P2E12.2' in the FORMAT statement.

Here is this code again printed with the corrections in place. One can access this corrected code on the Tutorial web page linked to the course web page by clicking on the second 'mycode.f' link.

```
123456789T123456...
      PROGRAM MYCODE
C
C This is my first Fortran code.
C
      INTEGER I, J, I_INDEX(100)
      REAL PI, LLCOEFF(100)
      REAL*8  DPI, SSCOEFF(100), BBOUT(100)
C
      PI = 3.14159
      DPI = 3.14159265359D0
C
C Calculate integer and single precision arrays.
C
      DO 52 I = 1, 100
         I_INDEX(I) = I*2
         LLCOEFF(I) = FLOAT(I**2) * SIN(PI*FLOAT(I)/10.)
  52  CONTINUE
C
C Calculate double precision arrays.  Fixed the label
C statement number for CONTINUE.
C
      DO 1005 J = 1, 100
         SSCOEFF(J) = COS(DPI*DFLOAT(J)/8.0D0)
         BBOUT(J) = SSCOEFF(J) * DEXP(-DFLOAT(J)/300.D0)
 1005 CONTINUE
C
C Output this data to the screen in ordered columns.
C
C Fixed error in maximum I, was 200, should be 100.
C
      DO 2344 I = 1, 100
```

```
         WRITE(*, 2340) I, I_INDEX(I), LLCOEFF(I), SSCOEFF(I),
     1      BBOUT(I)
C
C Fixed errors in FORMAT statement: F8.2 should be F10.2 and
C there should be a 2 in 1PE format code.
C
 2340    FORMAT(2X,2I5,2X,F10.4,2X,1P2E12.2)
 2344 CONTINUE
C
      STOP
      END
```

Feel free to play around with this code you made (by once again editing it in emacs). Try mistyping some of the lines in this code and see what compilation errors this may cause.