Some Useful Fortran Tricks

I'm not really a big fan of the word *tricks*, but rather here are some nice examples of things to do in Fortran that some may not find completely obvious.

1. Non-advancing status counter

```
PROGRAM progress
!Shows how to place a non-advancing status counter...
IMPLICIT NONE
INTEGER :: J, NR

NR=10

DO J=1,NR
  write(*,FMT="(A1,A,t21,F6.2,A)",ADVANCE="NO") achar(13), &
  & " Percent Complete: ", (real(J)/real(NR))*100.0, "%"

  CALL sleep(1)  !give a delay in sec to see the output

ENDDO

END PROGRAM progress
```

2. Converting an integer or real variable into a string

```
PROGRAM num2str
IMPLICIT NONE
REAL    :: num
INTEGER :: J
CHARACTER(LEN=3) :: istr1, istr2
CHARACTER(LEN=8) :: rstr1, rstr2
CHARACTER(LEN=100) :: outfile

J=10
write(istr1,"(I2.2)") J
write(istr2,"(I3.3)") J
write(*,*) istr1, istr2

num=13.875
write(rstr1,"(F8.2)") num
write(rstr2,"(F8.3)") num
write(*,*) trim(adjustl(rstr1)), " ", trim(adjustl(rstr2))

!demonstrate usefulness in naming multiple files
DO J=1,10
  write(istr1,"(I2.2)") J
  outfile = 'mydata_'//trim(istr1)//'.xyz'
  write(*,*) trim(outfile)
ENDDO

END PROGRAM
```

3. Reading command line arguments

Fortran programs can read command line arguments with the following two functions:

```
N = IARGC ( )          !sets N to the number of command line arguments

CALL GETARG(I, STR)  !Puts the i'th argument into string STR
```

```
PROGRAM commandline
IMPLICIT NONE
REAL :: num
INTEGER :: option2, N
CHARACTER(LEN=100) :: arg, option1

!Example to read a character and an integer from the command line.
!For example try,
! >> ./a.exe inputfilename 21

!Read input from the command line
N = IARGC()
IF (N < 1) THEN
 write(*,'(a)') "usage:  ./foo option1 option2"
 write(*,'(a)') "        option1=character"
 write(*,'(a)') "        option2=integer"
ELSE
 CALL GETARG(1,option1)      !Grab the first command line argument
                             ! and store it in the variable 'option1'

 CALL GETARG(2,arg)          !Grab the 2nd command line argument
                             ! and store it in the temporary variable
                             ! 'arg'

 read(arg,*) option2         !Now convert string to integer

  write(*,*) "Variable option1 = ", trim(adjustl(option1))
  write(*,*) "Variable option2 = ", option2
ENDIF

END PROGRAM commandline
```

4. Reading user variables

The subroutine **GETENV** can be used to retrieve environment variables.

```
PROGRAM getenviron
IMPLICIT NONE
CHARACTER(LEN=30)  :: user_name

CALL GETENV('USER', user_name)
write(*,*) user_name

END PROGRAM getenviron
```

5. Using Unix commands in your code

The **SYSTEM** subroutine is key to this one.

```
PROGRAM unixsystem
IMPLICIT NONE
CHARACTER(LEN=30) :: cmd    !string to store the Unix command

cmd = 'pwd'       ! As an example print the working directory

CALL SYSTEM(cmd)

END PROGRAM unixsystem
```

6. Reading input without having to know the number of lines in the file

This code example is not entirely bullet proof. It requires a parameter called maxrecs which is the maximum number of lines of a file you can read in. Two options exist (1) make this number larger than any file you can possibly imagine, or (2) if you exceed maxrecs then issue a error message and exit the code. Then recompile with a larger number of maxrecs.

```
PROGRAM readfile
IMPLICIT NONE
REAL, DIMENSION(:), ALLOCATABLE :: mydata
INTEGER, PARAMETER :: maxrecs = 10000
INTEGER :: J, NR, ios
CHARACTER(LEN=100) :: inputfile
CHARACTER(LEN=1)   :: junk

write(*,*) "Enter name of file to read in..."
read(*,*) inputfile

!Determine total number of lines in file
NR = 0
OPEN(UNIT=1,FILE=inputfile)
DO J=1,maxrecs
  READ(1,*,IOSTAT=ios) junk
  IF (ios /= 0) EXIT
  IF (J == maxrecs) THEN
    write(*,*) "Error:  Maximum number of records exceeded..."
    write(*,*) "Exiting program now..."
    STOP
  ENDIF
  NR = NR + 1
ENDDO
REWIND(1)

!Now we can allocate data variables
ALLOCATE(mydata(NR))

!Now read data into mydata
DO J=1,NR
  READ(1,*) mydata(J)
ENDDO
CLOSE(1)

END PROGRAM readfile
```