Controlling the program

In this section we will discuss several forms of possible control of the program e.g. input parameters, initial values of variables. Below are four sections dedicated to 1) Parsing command line arguments of the program, 2) Easy way of creating input files for the program, 3) Retriving enviromental variables from Unix shell, and the last one is a little disconnected 4) running Unix system command.

Command line arguments

In a command line environment arguments may be passed to the command which starts the program. The command line arguments may be obtain with a standard subroutine that is a part of Fortran 2003 i.e. `get_command_argument()` . The number of command line arguments may be obtained using the function `command_argument_count()` .

The subroutine `get_command_argument` has to be called with two arguments i.e. 1) number of the argument in the list, 2) string to hold the argument.

For example

```
program command_line_arguments

  implicit none

  integer :: narg
  integer :: i
  character(len=20) :: argument

  ! Get number of arguments given to the program
  narg = command_argument_count()

  print *,'Number of command line arguments : ', narg

  ! If any arguments present
  if(narg > 0) then

    ! Loop over the arguments, print number and value of the parameter
    do i=1,narg

      call get_command_argument(i,argument)
      print *,i,' -> ',argument

    end do

  end if

end program command_line_arguments
```

which may output

```
$ ./command_line.x --help --parameter one,two --list 34 --memory 0
 Number of command line arguments :            7
       1  -> --help
       2  -> --parameter
       3  -> one,two
       4  -> --list
       5  -> 34
       6  -> --memory
       7  -> 0
```

Namelist - input files prototype

Fortran has a construct which allows creating list of variables of different type and processing

their I/O collectively. The construct is called a `namelist` and is created

```
integer :: var1, var2
real :: var3, var4
namelist /unique_name/ var1, var2, var3, var4
```

`unique_name` is an arbitrary name which will characterize the namelist. The namelist can be read as one variable from a file or keyboard. An example usage may look like this

```
program namelist_input_file

  use iso_fortran_env

  implicit none

  integer :: momentum, nelectrons, charge
  real :: mass

  namelist /molecule/ momentum, mass, nelectrons, charge

  read(INPUT_UNIT, molecule)

  print molecule

end program namelist_input_file
```

and the input file

```
&molecule
  momentum = 0,
  mass = 123.0,
  nelectrons = 4,
  charge = -2,
/
```

the program can be now run as

```
$ ./namelist-example.x < input.dat
&MOLECULE
 MOMENTUM=          0,
 MASS=  123.000000    ,
 NELECTRONS=          4,
 CHARGE=         -2,
 /
```

Enviromental variables

Fortran 2003 defines a subroutine `get_environment_variable` for retrieving the value of Unix enviromental variable. The function is called with two arguments 1) name of the Unix variable, 2) string variable to hold the value.

Example program may look like this

```
program enviroment

  implicit none
  character(len=100) :: val

  call get_environment_variable('HOME',val)
  print *,'HOME -> ',trim(val)

  call get_environment_variable('USER',val)
  print *,'USER -> ',trim(val)

  call get_environment_variable('SHELL',val)
  print *,'SHELL -> ',trim(val)

  call get_environment_variable('LANG',val)
  print *,'LANG -> ',trim(val)

end program enviroment
```

which outputs

```
HOME -> /Users/marcin
USER -> marcin
SHELL -> /bin/bash
LANG -> en_US
```

Executing system command from Fortran program

Fortran 2008 defines a standard subroutine for executing system commands. The build in subroutine is `execute_command_line` . Example usage may look like

```
program system_command

  implicit none

  integer :: error

  call execute_command_line ("ls -l", exitstat=error)

end program system_command
```

which will call the `ls -l` command in the work directory and print the output on the screen.