Programming and Modelling (week 36) b

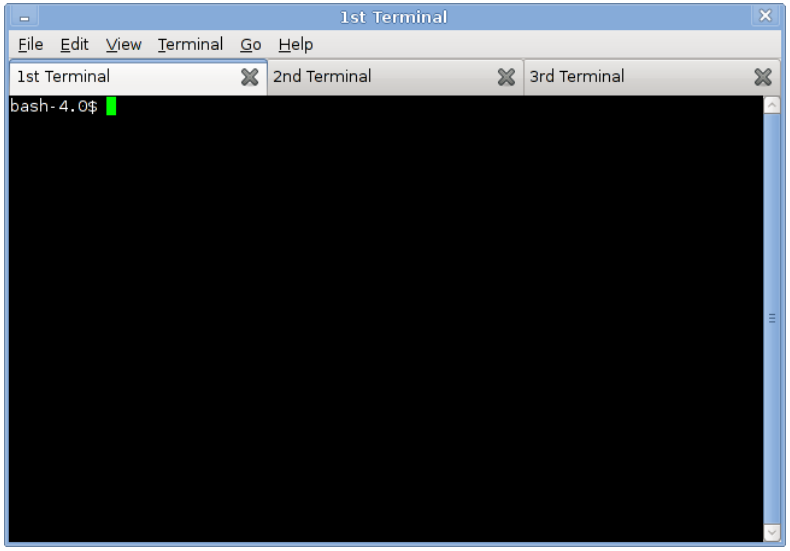
C. Thieulot

Institute of Earth Sciences

September 2017

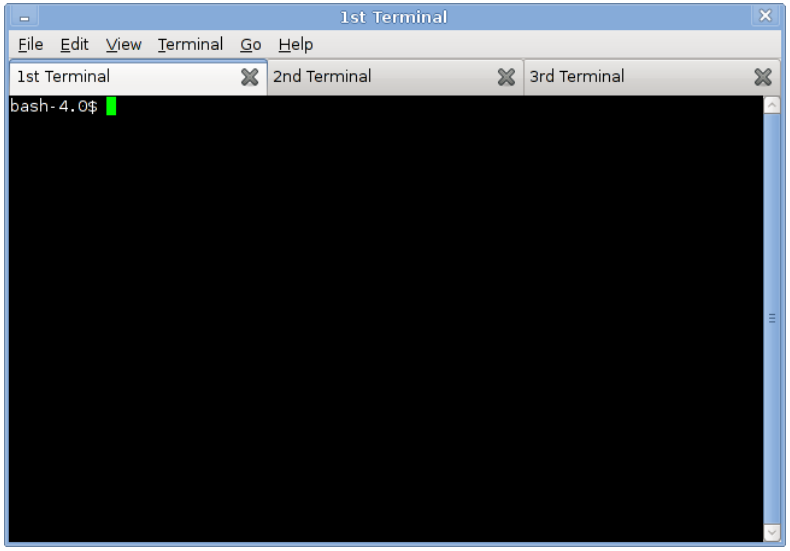
Shell Commands (1)

We are going to use a terminal:



Shell Commands (1)

We are going to use a terminal:



Shell commands (2)

- The **shell** is a program that takes your commands from the keyboard and gives them to the operating system to perform.
- In the old days, it was the only user interface available on a Unix computer.
- We need to learn a few commands:
  - `mkdir` (make directory)
  - `cd` (change directory)
  - `ls` (list)
  - `rm` (remove)

Fortran (1)

- Fortran
  (IBM Mathematical Formula Translating System)
- a general-purpose language especially suited to numeric computation and scientific computing.
- Originally developed by IBM at their campus in south San Jose, California in the 1950s.
- has been in continual use for over half a century in computationally intensive areas such as numerical weather prediction, finite element analysis, computational fluid dynamics, computational physics and computational chemistry.
- It is one of the most popular languages in the area of high-performance computing and is the language used for programs that benchmark and rank the world's fastest supercomputers. (www.top500.org)

Fortran (2)

- FORTRAN (1957)
- FORTRAN II (1958)
- FORTRAN IV (1962)
- FORTRAN 66
- FORTRAN 77 (standardised)
- FORTRAN 90 
- FORTRAN 95
- FORTRAN 2003
- FORTRAN 2008

A very basic program (1)

- We create a file *myfirstprogram.f90*
- It looks like

```
program compute_smthg
implicit none



 set of instructions



end program
```

- We compile it

  `> gfortran myfirstprogram.f90`

- this create the file *a.out* which we execute:

  `> ./a.out`

A very basic program (2)

```
program lmfao_print
implicit none


write(*,*) 'look at that body'
write(*,*) 'I work out'


end program
```

Numbers

- In mathematics: integers, irrationals, complex numbers ...
  $\mathbb{P}$, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{D}$, $\mathbb{I}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$

Numbers

- In mathematics: integers, irrationals, complex numbers ...
  $\mathbb{P}$, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{D}$, $\mathbb{I}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$
- There are six basic Fortran data types:
  - `INTEGER`, -2147483648 to 2147483647
  - `REAL` 1.175495×10^{-39} to 3.402823×10^{39}
    precision confined to first 7 decimal digits
  - `DOUBLE PRECISION`, also `REAL(8)`
    precision confined to first 15 decimal digits
  - `LOGICAL`, Boolean values (true or false)
  - `CHARACTER`
  - `COMPLEX`

Example (1)

Let us write a simple program which converts Fahrenheit degrees in Celsius degrees.

Example (1)

Let us write a simple program which converts Fahrenheit degrees in Celsius degrees.

$$T_C = \frac{5}{9}[T_F - 32]$$

Example (1)

Let us write a simple program which converts Fahrenheit degrees in Celsius degrees.

$$T_C = \frac{5}{9}[T_F - 32]$$

The fortran file is named *convert_F_to_C.f90* and looks like this:

```
program F_to_C
implicit none
real ::  temp_in_F
real ::  temp_in_C
temp_in_F=75.1
temp_in_C=(temp_in_F-32.)*5./9.
write(*,*) 'Temp in F=',temp_in_F
write(*,*) 'Temp in C=',temp_in_C
end program
```

Example (2)

```
> gfortran convert_F_to_C.f90
> ./a.out
Temp in F= 75.099998
Temp in C= 23.944445
```

Example (3)

Problem: if I change the value of `temp_in_F`, I need to recompile and run the program...
$\rightarrow$ I wish for the program to ask me for a temperature in F and give me its equivalent in C.

Example (3)

Problem: if I change the value of `temp_in_F`, I need to recompile and run the program...
$\rightarrow$ I wish for the program to ask me for a temperature in F and give me its equivalent in C.
The fortran file is named *convert_F_to_C_2.f90* and looks like this:

```
program F_to_C
implicit none
real ::  temp_in_F
real ::  temp_in_C
read(5,*) temp_in_F
temp_in_C=(temp_in_F-32.)*5./9.
write(*,*) 'Temp in F=',temp_in_F
write(*,*) 'Temp in C=',temp_in_C
end program
```

'5' is the unit of the stdin

Example (4)

```
thebeast:progmod geogarfield$ ./a.out
12.1
 Temp in F=   12.100000
 Temp in C=  -11.055555
thebeast:progmod geogarfield$ ./a.out
45.2
 Temp in F=   45.200001
 Temp in C=   7.3333335
thebeast:progmod geogarfield$ ./a.out
85.1547
 Temp in F=   85.154701
 Temp in C=   29.530388
thebeast:progmod geogarfield$ ./a.out
0.
 Temp in F=   0.0000000
 Temp in C=  -17.777779
thebeast:progmod geogarfield$ ./a.out
100.
 Temp in F=   100.00000
 Temp in C=   37.777779
```

Arrays (1)

Problem: what if I wanted to convert 5 temperatures at once ?

Solution:

```
program F_to_C
implicit none
real ::  temp_in_F1
real ::  temp_in_F2
real ::  temp_in_F3
real ::  temp_in_F4
real ::  temp_in_F5
real ::  temp_in_C1
real ::  temp_in_C2
real ::  temp_in_C3
etc ...
```

Arrays (1)

Problem: what if I wanted to convert 5 temperatures at once ?

Solution:

```
program F_to_C
implicit none
real ::  temp_in_F1
real ::  temp_in_F2
real ::  temp_in_F3
real ::  temp_in_F4
real ::  temp_in_F5
real ::  temp_in_C1
real ::  temp_in_C2
real ::  temp_in_C3
etc ...
```

$\Rightarrow$ what if I have 1000 temperature measurements ?

Arrays (2)

Better solution: declare an array of values at once !

```
program F_to_C
implicit none
real, dimension(10) ::  temp_in_F
real, dimension(10) ::  temp_in_C
etc ...
```

do loops (1)

Q: Now that I have an empty array,
how can I fill it up with values ?

do loops (1)

Q: Now that I have an empty array,
how can I fill it up with values ?

By hand ?

```
temp_in_F(1)=1.12
temp_in_F(2)=4.85
temp_in_F(3)=9.54
temp_in_F(4)=12.5
...
temp_in_F(1000)=0.5
```

$\rightarrow$ I need to find a more systematic way: the **do loop**

do loops (2)

file *myfirstloop.f90*

```
program loop1
implicit none
integer ::  i
do i=1,5
   write(*,*) 'hello',i
end do
end program
```

do loops (2)

file *myfirstloop.f90*

```
program loop1
implicit none
integer ::  i
do i=1,5
   write(*,*) 'hello',i
end do
end program
```

```
> ./a.out
hello 1
hello 2
hello 3
hello 4
hello 5
```

do loops (3)

file *mysecondloop.f90*

```
program loop2
implicit none
integer ::  i, n, sum
n = 10
sum = 0
do i = 1,n
   sum = sum + i
   write(*,*) 'i =',i,'; sum =', sum
end do
end program
```

do loops (3)

file *mysecondloop.f90*

```
program loop2
implicit none
integer ::  i, n, sum
n = 10
sum = 0
do i = 1,n
   sum = sum + i
   write(*,*) 'i =',i,'; sum =', sum
end do
end program
```

```
> ./a.out
i = 1 ; sum = 1
i = 2 ; sum = 3
i = 3 ; sum = 6
i = 4 ; sum = 10
i = 5 ; sum = 15
i = 6 ; sum = 21
```

do loops (4)

```
program F_to_C
implicit none
integer ::  i
real, dimension(10) ::  temp_in_F
real, dimension(10) ::  temp_in_C
do i=1,10
   temp_in_F(i)=i*10.
end do
do i=1,10
   temp_in_C(i)=(temp_in_F(i)-32.)*5./9.
end do
do i=1,10
   write(*,*) 'Temp in F=',temp_in_F(i), &
          ' -> in C=',temp_in_C(i)
end do
end program
```

do loops (4)

```
> ./a.out
Temp in F= 10.000000 -> in C= -12.222222
Temp in F= 20.000000 -> in C= -6.6666665
Temp in F= 30.000000 -> in C= -1.1111112
Temp in F= 40.000000 -> in C= 4.4444447
Temp in F= 50.000000 -> in C= 10.000000
Temp in F= 60.000000 -> in C= 15.555555
Temp in F= 70.000000 -> in C= 21.111111
Temp in F= 80.000000 -> in C= 26.666666
Temp in F= 90.000000 -> in C= 32.222221
Temp in F= 100.00000 -> in C= 37.777779
```

write

Q: How do I write my results in a file ?

write

Q: How do I write my results in a file ?

```
program oplaopla
...
open(unit=123,file='results.dat', &
                action='write', &
                status='replace')
do i=1,100
   write(123,*) i+1
end do
close(123)
...
end program
```

DO's and DON'Ts (1)

1) write your program piece by piece, compile and test often
2) indent your loops (3 steps):

```
do i=1,10

   do j=1,100

      do k=-2:2

         ...
         ...
         array(i,j,k)=i+j-k
         ...
         ...

      end do

   end do

end do
```

DO's and DON'Ts (2)

3) comment your program

```
do ic=1,ncell
   inoode(1)=grid%icon(1,ic)
   inoode(2)=grid%icon(2,ic)
   inoode(3)=grid%icon(3,ic)
   inoode(4)=grid%icon(4,ic)
   !=====[loop over 4 integration points]=====
   do iq=1,4
      counterq=counterq+1
      if (curved) then
         theta=atan(grid%xq(counterq)/grid%yq(counterq))
      end if
      do i=1,4
         i1=2*i-1
         i2=2*i
         Bmat2D(1,i1)=dNdx2D(i) ; Bmat2D(1,i2)=0.d0
         Bmat2D(2,i1)=0.d0      ; Bmat2D(2,i2)=dNdy2D(i)
         Bmat2D(3,i1)=dNdy2D(i) ; Bmat2D(3,i2)=dNdx2D(i)
      end do
   end do
   !=====[penalty term integration point]=====
   iq=5
   do i=1,4
      i1=2*i-1
      i2=2*i
      Bmat2D(1,i1)=dNdx2D(i) ; Bmat2D(1,i2)=0.d0
      Bmat2D(2,i1)=0.d0      ; Bmat2D(2,i2)=dNdy2D(i)
      Bmat2D(3,i1)=dNdy2D(i) ; Bmat2D(3,i2)=dNdx2D(i)
   end do
   !=====[open boundary conditions]=====
   ...
```

DO's and DON'Ts (3)

template.f90 (on my website)

```
!============================================================!
!
! student number(s):
!
! date:
!
! exercise number:
!
! description of the program:
!
!
!
! comments to corrector:
!
!
!============================================================!

program ....




end program

!============================================================!
```