Basics of Programming on Linux

1. Installation of Ubuntu Linux

   - Download the image of Ubuntu from the following web page: http://www.ubuntu.com/getubuntu/download
   - Burn it onto a blank CD (or DVD).
   - Backup your important files.
   - Insert it in the CD (or DVD)-ROM and reboot the computer. [Check if your computer (the BIOS) is set to boot from the CD-ROM drive.]
   - Follow the instruction that will appear on the screen to install Ubuntu.

   After successful installation of Ubuntu,

   - Find "Application" in the left corner of the desktop. Click it and go to "Accessories." Find "Terminal" to start up.
   - Type command in terminal to start up the software, such as gfortran, gnuplot, etc.
   - If the software is not installed, you will be asked to type "sudo apt-get install …" Follow the instruction. If it does not work, type "sudo apt-get update" or "sudo apt-get upgrade" to update the operating system. Then, install the software.

   If you are trouble with USM campus wireless, go at https://wireless.usm.edu/xpressHelp/ubuntulinux/ubuntulinux.html

2. Confirmation whether compilers are installed

Start up the terminal and type the following commands to see if these are installed.

Compilers	Commands to compile	Execution
Fortran	gfortran (or f77, g77, f90, etc.)	./compiledfile
C	gcc	./compiledfile
C++	g++	./compiledfile
Java	javac	java filename

e.g.

Fortran	C
> gfortran samplecode.f > ./a.out	> gcc samplecode.c > ./a.out
> gfortran –o sample.out samplecode.f > ./sample.out	> gcc sample.out samplecode.c > ./sample.out

Note: For "gcc", if you use mathematical functions, you have to use the option, –lm. For example, "gcc example.c –lm"

3. Fortran and C commands

Input (the example codes):

Fortran	C

`    READ(5,10) A`
`10  FORMAT (F10.5)`
`    STOP`
`    END`
	`void main() {`
	`    scanf("%lf",&d);`
	`}`

Output (the example codes):

Fortran	C

`      WRITE(6,10)`
`   10 FORMAT (' FORTRAN-PROGRAM')`
`      STOP`
`      END`
	`void main() {`
	`     printf("C-PROGRAM\n");`
	`}`

`WRITE(*,*)' FORTRAN-PROGRAM'`
`      STOP`
`      END`	

Comment:

Fortran	C
At the first column: type like "c."	/* "comments" */
Anywhere type "!" then type any comments for the line.	// "comments"

Declaration of data types (the commands):

Fortran	C	Examples of the number
INTEGER*2	short int	3
INTEGER	int (or long)	1726549
REAL	float	3.87
REAL*8 (or DOUBLE PRECISION)	double	4.7D+4
CHARACTER*n	char[n]	'Hello'
COMPLEX*8	float _Complex	2-3.8I
COMPLEX*16	double _Complex	
LOGICAL		

For character, *n* is an integer. For C, you must use #include <complex.h> as one of the headers. To compile a C program with complex variables, use "gcc -lm -o a.out program.c"

Variable restrictions:

Fortran	C
Alphabets, numbers, and underscore	Alphabets, numbers, and underscore
Must start with an alphabet.	Must start with an alphabet or an underscore.
8 letters (for an old Fortran)	31 letters

Note: The above restrictions depend on the compilers. Some of them are case sensitive.

Omission of the variable declaration:

This feature is only for Fortran. For C, all of the variables have to be declared. Without the following declarations in the table, the compiler will recognize as follows:

- If the first letter of a variable is A, B, C, D, E, F, G, H, O, P, Q, R, S, T, U, V, W, X, Y, and Z, the variable will be automatically assigned as real.
- If the first letter of a variable is I, J, K, L, M and N, the variable will be automatically assigned as integer.

Fortran commands	Usages
IMPLICIT NONE	Deactivate the above features.
IMPLICIT REAL*8 (A-H,O-Z)	All of the variable used in the program, whose first letter starts with A to H and O to Z, will be assigned as double precision. The others will be assigned as integer.

Array:

Fortran	C
REAL DIMENSION A(3,4)	float A[3][4];

REAL DIMENSION A(3,4)
Each slot in the memory storage for Fortran:

A(1,1)	A(2,1)	A(3,1)
A(1,2)	A(2,2)	A(3,2)
A(1,3)	A(2,3)	A(3,3)
A(1,4)	A(2,4)	A(3,4)

float A[3][4];
Each slot in the memory storage for C:

A[0][0]	A[0][1]	A[0][2]	A[0][3]
A[1][0]	A[1][1]	A[1][2]	A[1][3]
A[2][0]	A[2][1]	A[2][2]	A[2][3]

Initial value setting:

Fortran	C
REAL A REAL B DATA A/45.9/ B/11.98/	float A = 45.9; float B = 11.98;
REAL C(3) DATA C/1.0, 2.5, 7.8/	float c[3] = {1.0,2.5,7.8};

Sharing of storage units:

Fortran	C
INTEGER A REAL B EQUIVALENCE(A,B)	Union UT{ int A; float B; }UN;

Mathematical operators:

	Fortran	C
Addition or identity	+	+
Deduction or negation	-	-
Multiplication	*	*
Division	/	/
Exponentiation	**	Pow(2,6) [$=2^6$]

Relational expressions:

	Fortran	C
Less than	< or .LT.	<

Less than or equal to	<= or .LE.	<=
Equal to	= = or .EQ.	= =
Not equal to	/= or .NE.	!=
Greater than	> or .GT.	>
Greater than or equal to	>= or .GE.	>=

Logical expressions:

	Fortran	**C**
Negation	.NOT.	!
Conjunction	.AND.	&&
Inclusive disjunction	.OR.	||
Equivalence	.EQV.	= =
Nonequivalence	.NEQV.	!=
Exclusive disjunction	.XOR.	^

Control statements:

- Go To

	Fortran	**C**

Unconditional	10 WRITE 'Come here.'
	
	GOTO 10
		flag: printf("Come here.\n");
		
		if(a != 0){ goto flag;}

Computed [When the value of N is 1, go to line 10. When N is 2, go to 20, and etc.]	GOTO (10, 20, 30, 40) N	

- If

	Fortran	**C**
One condition	IF (A.GT.0.0) A=10.0	if (A > 0.0){ A = 10.0;}

More than one conditions	IF (A.GT.0.0) THEN A = A +1.0
	ELSE IF (A.GT.10.0) THEN
	A = A + 2.0
	ELSE
	A = 0.0
	END IF
		if (A>0.0){A=A+1.0;}
		else if (A>10.0){A=A+2.0;}
		else {A=0.0;}

Arithmetic conditions [If A is less than 0, go to S1. If A is equal to 0, go to S2. If A is greater than 0, go to S3.]	IF (A) S1, S2, S3 (This is aborted since Fortran95.)	
Logical conditions [If A is true, execute ST.]	LOGICAL A If (A) ST	

- Loop statements (Do or For)

	Fortran	C

The basic loop calculation	DO 10 I = 1, 50, 2
	A=A+I
	WRITE(*,*) 'A=', A
	10 CONTINUE
		for (i=1;i<=50;i=i+2){
		a=a+I;
		printf("a=%d\n",a);
		}

	DO WHILE(I<11)
	J=I
	MULTI=MULTI*A
	I=I+1
	END DO
		while(i<11){
		j=I;
		multi=multi*a;
		i=i+1;
		}

Try this!

Fortran	C

Program volume of sphere
 	
integer ii
real*8 pi,rr,vol
 	
open(6,file='volume.fdata',status='unknown')
 	
pi = 3.1415926535d0
rr = 0.0d0
vol = 0.0d0
do ii = 1, 100
vol = 4.0d0/3.0d0 * pi * rr ** 3
write(6,*) ii, rr, vol
rr = rr + 0.1d0
end do
stop
end
	`#include<stdio.h>`
	`#include<math.h>`
	
	`int main(){`
	`FILE *fp;`
	
	`int i;`
	`double pi,rr,vol;`
	
	`pi = 3.1415926535;`
	`fp = fopen("volume.cdata","w");`
	
	`for(i=1;i<=100;i++){`
	`vol = 4.0/3.0 * pi * pow(rr,3);`
	`fprintf(fp, "%i %f %f\n", i, rr, vol);`
	`rr = rr + 0.1;`
	`}`
	`fclose(fp);`
	`return 0;`
	`}`

Compile the programs and execute them to have the data file, volume.fdata and volume.cdta.

Notes:

- In the Fortran code, 4.0d0 means 4.0×10^0 with double precision. If you type like 4.0d3, this means 4.0×10^3.
- In the C code, there is a print command, "`fprintf(fp, "%i %f %f\n", i, rr, vol);`" "%i" and "%f" specify the type of output values. %i and %f are integer and float, respectively. The command, \n, is for starting the new line. (Be careful! It is a backslash not a slash!)
- For most of the programming languages, only one equal expresses input a value in a specified variable. "rr=0.0" means that 0.0 is substituted in the variable rr. Therefore, "rr=rr+0.1" explains that the new rr is substituted by the old rr plus 0.1.