Chapter: 2

File Opening and Creation

Introduction :

The "Open File" chapter in Fortran covers a crucial aspect of programming, which is opening and manipulating files. In this chapter, we explore how to open files in the Fortran programming language, which is commonly used for scientific and numerical computations.

Opening a file allows a Fortran program to access the data stored in that file, whether it is for reading or writing. The "Open File" chapter introduces the syntax of the "open" statement in Fortran, which is used to open a file and associate it with a file variable in the program.

Throughout the chapter, you will learn about the different parameters used when opening a file in Fortran. This includes the file unit number, which is used to identify the file in the program, as well as options such as the file status (new, existing), the action to be taken on the file (read, write), and the file access mode (sequential, direct).

The chapter also provides concrete examples to illustrate how to open a file in Fortran and how to use the different available options. You will learn how to specify the file name, handle file opening errors, and perform read and write operations on opened files.

In summary, the "Open File" chapter in Fortran is a valuable resource for programmers looking to master file manipulation in this language. It provides a clear introduction to the syntax and options for opening files in Fortran, allowing you to fully leverage the file manipulation capabilities of this powerful programming language.

Using a file within a program requires opening it. In Fortran, the open instruction is used.

Exemples :

- Open (10, file='result.dat') ! Compact form
- Open (10, file='result.dat',status='old') ! Form with option

Syntaxe OPEN

The OPEN syntax in Fortran is used to open files and associate them with file variables to enable reading and writing data to those files.

```
OPEN(unit = unit_number, file = file_name, status = file_status, action = file_action,
access = file_access, iostat = error_code)
-------------------------------------------!
Close(unit = unit_number)
```

In this syntax:

- **unit_number:** is the unit number used to identify the file within the program.
- **file_name**\: is the name of the file to be opened.
- **file_status**: specifies the status of the file, such as "OLD", "NEW", or "REPLACE".
- **file_action** : specifies the action to be taken on the file, such as "READ", "WRITE", or "READWRITE".
- **file_access** : specifies the access mode for the file, such as "SEQUENTIAL" or "DIRECT".
- **error_code**: is an optional variable that stores the error code generated during the file

Example: To calculate the sum of two matrices A and B and store the results in a file on the desktop, for example

```
PROGRAM sum
IMPLICIT NONE
INTEGER :: i, j
REAL :: A(3,3), B(3,3), C(3,3)
INTEGER :: lun
lun = 20
OPEN(unit=lun, file="resultats.TXT", status="REPLACE")
A = reshape((/1., 2., 3., 4., 5., 6., 7., 8., 9./), (/3, 3/))
B = reshape((/9., 8., 7., 6., 5., 4., 3., 2., 1./), (/3, 3/))
C = A + B
WRITE(lun,*) 'Matrice A :'
DO i = 1, 3
WRITE(lun, '(3(F5.1, 1X))') A(i, :)
END DO
WRITE(lun,*) 'Matrice B :'
DO i = 1, 3
WRITE(lun, '(3(F5.1, 1X))') B(i, :)
END DO
WRITE(lun,*) 'Somme C :'
DO i = 1, 3
WRITE(lun, '(3(F5.1, 1X))') C(i, :)
END DO
CLOSE(lun)
END
```

- The simplified syntax for the "open" statement in Fortran, along with the corresponding "close" statement:

```
OPEN(unit_number, file=file_name, status=file_status)
! Perform file operations...

CLOSE(unit_number)
```

Explanation of the parameters:

- **unit_number**: The unit number that identifies the file within the program.
- **file=file_name**: The name of the file to be opened.
- **status=file_status**: The status of the file, such as "OLD," "NEW," or "REPLACE."

After performing the necessary file operations, you can close the file using the "close" statement. The "close" statement is used to indicate the end of file operations for a specific file unit.

Make sure to replace unit_number, file_name, and file_status with the appropriate values based on your requirements.

There are options for the "status" parameter in the "open" command, such as 'new' to create a new file (and generate an error if the file already exists), 'old' to open an existing file (and generate an error if the file does not exist), 'scratch' to create a temporary file, etc.

• The "open" command with the status='replace' parameter, and you name your file data.txt. If you run the program multiple times, each time you open the data.txt file, the previous content of the file will be erased and replaced with the new content generated by your program.
• This can be useful if you want to save the results of your program in a file and do not need to keep the old data in the file. If you instead want to add data to an existing file, you can use the status='append' parameter in the open command. This way, the new data will be added to the end of the file without erasing the previous content.

Here's a brief explanation of the other options for the status parameter in the open command:
• '**new**': Used to create a new file. If the file already exists, an error will be generated.
• '**old**': Used to open an existing file. If the file does not exist, an error will be generated.
• '**replace**': Used to open a file in replacement mode. If a file with the same name already exists, it will be replaced by the newly created file. If the file does not exist, it will be created.
• '**append**': Used to add data to the end of an existing file. If the file does not exist, it will be created.
• '**scratch**': Used to create a temporary file that will be automatically deleted at the end of the program.

• '**unknown**': Used to open a file whose name is unknown at compilation. This option is typically used for dynamic file input/output, such as files generated by another program.