Lecture-10

File Processing

Lubna Ahmed

Introduction

- Programs shown up to this point had involved relatively small amount of input/output data.
- Input data were read from a terminal and output was displayed in the terminal.
- This is adequate if the volume of data involved is not large.
- Applications involving large data sets can be processed more conveniently if the data is stored in files.

File Basics

- Computers have a standard structure **for holding data** that can be accessed as a unit. This structure is called a **file**.
- A file consists of many lines of related data that can be accessed as a unit. **Each line of information in a file** is called a **record**.
- Fortran can read information from a file or write information to a file one record at a time.

Fortran Control Characters

I/O Statement	Function
OPEN	Associate a specific disk file with a specific I/O unit number
CLOSE	End the association of a specific disk file with a specific I/O unit number
READ	Read data from a specified I/O unit number
WRITE	Write data to a specified I/O unit number
REWIND	Move to the beginning of a file
BACKSPACE	Move back one record in a file

Fortran Control Characters cont'd

Opening and closing a file

Before you can use a file you have to open it. The command is

OPEN (list-of-specifies)

where the most common specifies are:

UNIT= associate unit number
FILE = file name
ACCESS = data input mode
STATUS = file type Old or New

Fortran Control Characters cont'd

When you are done with the file, it should be closed by the statement

CLOSE (UNIT)

Read and write revisited

he only necessary change from our previous simplified READ/WRITE statements, is that the unit number must be specified.

READ(UNIT,*) variable to read data

WRITE(UNIT,*) data

Unit Specifier

The unit specifier has the following form:

UNIT = *integer-expression*

where the value of integer-expression is a non-negative integer that designates the unit number to be connected to this file. Reference to this file by subsequent READ or WRITE statements is by means of this unit number.

FILE = Clause

The FILE = clause has the form:

FILE = *character-expression*

where the value of character-expression is the name of the file to be connected to the specified unit number.

STATUS = Clause

The STATUS = Clause has the form:

STATUS = *character-expression*

where the value of character-expression is one of the following:

OLD

NEW

REPLACE

SCRATCH

STATUS = Clause cont'd

- **OLD** means that the file already exists in the system.
- **NEW** means that the file does not yet exist and is being created by the program
- **REPLACE** creates a fresh file and deletes any old file of the same name.
- **SCRATCH** is used for a file that has not been given a name, creating a "scratch" file for temporary use as the program executes: a scratch file is deleted when the program terminates or when a CLOSE statement (below) is executed for the unit.

ERR = Clause

The ERR = clause has the form:

ERR = *n*

where *n* is the label of an executable statement that is the next statement executed if an error occurs in attempting to open the file.

IOSTAT = Clause

The IOSTAT = clause has the form:

IOSTAT = status-variable

where status-variable is an integer variable to which a value of 0 is assigned if the file is open successfully.

Opening Files

Before a file can be used for input or output, it must be opened by using an open statement of the form:

OPEN (open-list)

where open-list must include:

1. A unit specifier indicating a unit number to be connected to the file
2. A FILE = clause giving the name of the file being opened
3. A STATUS = clause specifying whether the file is old or new

It may also include:

4. An IOSTAT = clause indicating whether the file has been successfully opened
5. An ERR = clause specifying a statement to be executed if an error occurs while attempting to open the file

Closing Files

The CLOSE statement is of the form:

CLOSE (close-list)

where close list must include

1. A unit specifier

It may also include:

2. An IOSTAT = clause
3. An ERR = clause
4. A STATUS = clause specifying whether the file is to be kept or deleted. It has the form

   STATUS = character-expression

   where value of the character-expression is

   KEEP or DELETE

File Input

Data can be read from a file using a READ statement of the general form:

READ (control-list) input-list

where control-list must include:

1. A unit specifier indicating the unit connected to the file

It may also include one or more of the following:

2. A format specifier describing the format of the information to be read
3. An END = clause specifying a statement to be executed when the end of a file is reached
4. An ERR = clause specifying a statement to be executed if an input error occurs.
5. An IOSTAT = clause to check the status of input operation

File Output

Data are written to a file using a WRITE statement of the general form

WRITE (control-list) output-list

The control-list must include:

1. A unit specifier indicating the unit number connected to the file

It may also include one or more of the following:

2. A format specifier
3. An ERR = clause
4. An IOSTAT = clause

File Positioning Statement

- Ordinary Fortran files are sequential. However, we sometimes need to read a piece of data more than once or to process a file more than once during a program.
- Fortran provides two statements to help us move around within a sequential file.

1. **REWIND (position-list)**

This statement restarts the file at its beginning. It has the form:

```
REWIND(UNIT= unit)
```

where unit is the I/O unit number associated with the file that we want to work with

File Positioning Statement cont'd

2. BACKSPACE (position-list)

This statement positions the file at the beginning of the preceding record, i.e. **this statement moves back one record each time it is called.**

```
BACKSPACE(UNIT= unit)
```

where unit is the I/O unit number associated with the file that we want to work with

Case 1: Opening a file for input

The following statement opens a file named EXAMPLE.DAT and attaches it to I/O unit 8.

```
Integer::ierror
OPEN (UNIT=8,
      FILE='EXAMPLE.DAT',STATUS='OLD',ACTION=
      'READ', IOSTAT=ierror)
```

Case 1: Opening a file for input cont'd

- The STATUS='OLD' clause specifies that the file already exists;
- If it does not exist, then the OPEN statement will return an error code in variable ierror.
- This statement illustrates the proper form of the OPEN statement for an input file.
- The ACTION='READ ' clause specifies that the file should be read only.
- If an attempt is made to write to the file, an error will occur. This behavior is appropriate for an input file.

Case 2: Opening a file for output

The following statement opens a file named OUTDAT and attaches it to I/O unit 25.

```
Integer::unit, ierror
CHARACTER(len=6)::filename
unit=25
OPEN (UNIT=unit, FILE=filename,
      STATUS='NEW',ACTION= 'WRITE',
      IOSTAT=ierror)
```

Case 2: Opening a file for output cont'd

- The STATUS='NEW' clause specifies that the file is a new file.
- If it already exists, then the OPEN statement will return an error code in variable ierror.
- The ACTION='WRITE' clause specifies that the file should be write only.
- If an attempt is made to read from the file, an error will occur. This behavior is appropriate for an output file.

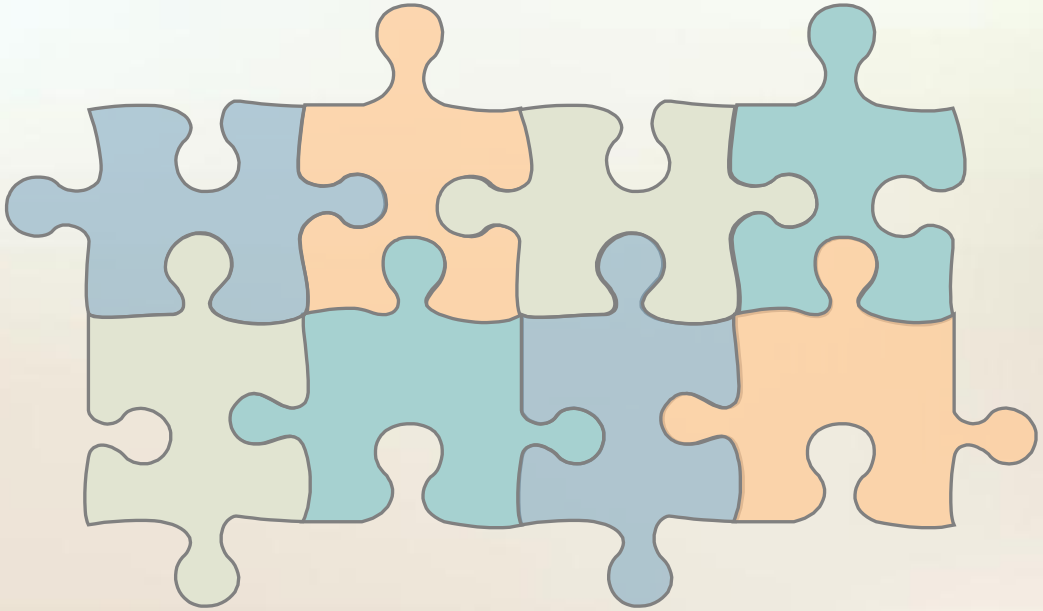
Case 3: Opening a Scratch file

The following statement opens a scratch file and attaches it to I/O unit 12.

```
OPEN (UNIT=12, STATUS='SCRATCH',
      IOSTAT=ierror)
```

- A scratch file is a temporary file that is created by the program and that is deleted automatically when the file is closed or when the program terminates.
- This type of file may be used for saving intermediate results while a program is running.
- No file name is specified in the OPEN statement. In fact , it is an error to specify a file name with a scratch file.
- The absence of ACTION = clause indicates that the file has been opened for both reading and writing.

Thanks



Given a noisy set of measurements(x,y)that appear to fall along a straight line, how can we find the equation of the line,

$y=mx+b$ ……(1)

that best fits the measurement? If we can determine the regression coefficients m and b, then we can use this equation to predict the values of y at any given x by evaluating equation (1) for that value of x. The slope of the least square line is given by

$$m = \frac{(\sum xy) - (\sum x)\bar{y}}{(\sum x^2) - (\sum x)\,\bar{x}}$$

and the intercept is given by $b = \bar{y} - m\bar{x}$

Write a program that will calculate the least square slope and intercept for a given set of noisy measured data points(x,y) that are to be found in an input data file.