COBOL - FILE HANDLING VERBS

File handling verbs are used to perform various operations on files. Following are the file handling verbs:

- Open
- Read
- Write
- Rewrite
- Delete
- Start
- Close

Open Verb

Open is the first file operation that must be performed. If Open is successful, then only further operations are possible on a file. Only after opening a file, the variables in the file structure are available for processing. **FILE STATUS** variable is updated after each file operation.

Syntax

```
OPEN "mode" file-name.
```

Here, file-name is string literal, which you will use to name your file. A file can be opened in the followin modes:

Mode	Description
Input	Input mode is used for existing files. In this mode, we can only read the file, no other operations are allowed on the file.
Output	Output mode is used to insert records in files. If a **sequential file** is used and the file is holding some records, then the existing records will be deleted first and then new records will be inserted in the file. It will not happen so in case of **indexed file** or **relative file.**
Extend	Extend mode is used to append records in a **sequential file**. In this mode, records are inserted at the end. If file access mode is **Random** or **Dynamic**, then extend mode cannot be used.
I-O	Input-Output mode is used to read and rewrite the records of a file.

Read Verb

Read verb is used to read the file records. The function of read is to fetch records from a file. At each read verb, only one record can be read into the file structure. To perform a read operation, open the file in INPUT or I-O mode. At each read statement, the file pointer is incremented and hence the successive records are read.

Syntax

Following is the syntax to read the records when the file access mode is sequential:

```
READ file-name NEXT RECORD INTO ws-file-structure
   AT END DISPLAY 'End of File'
```

```
   NOT AT END DISPLAY 'Record Details:' ws-file-structure
END-READ.
```

Following are the parameters used:

- NEXT RECORD is optional and is specified when an indexed sequential file is being read sequentially.
- INTO clause is optional. ws-file-structure is defined in the Working-Storage Section to get the values from the READ statement.
- AT END condition becomes True when the end of file is reached.

ExampleThe following example reads an existing file using line sequential organization. This program can be compiled and executed using **Try it** option where it will display all the records present in the file.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

ENVIRONMENT DIVISION.
   INPUT-OUTPUT SECTION.
      FILE-CONTROL.
      SELECT STUDENT ASSIGN TO 'input.txt'
      ORGANIZATION IS LINE SEQUENTIAL.

DATA DIVISION.
   FILE SECTION.
   FD STUDENT.
   01 STUDENT-FILE.
      05 STUDENT-ID PIC 9(5).
      05 NAME PIC A(25).

   WORKING-STORAGE SECTION.
   01 WS-STUDENT.
      05 WS-STUDENT-ID PIC 9(5).
      05 WS-NAME PIC A(25).
   01 WS-EOF PIC A(1).

PROCEDURE DIVISION.
   OPEN INPUT STUDENT.
      PERFORM UNTIL WS-EOF='Y'
         READ STUDENT INTO WS-STUDENT
            AT END MOVE 'Y' TO WS-EOF
            NOT AT END DISPLAY WS-STUDENT
         END-READ
      END-PERFORM.
   CLOSE STUDENT.
STOP RUN.
```

Suppose the input file data available in the **input.txt** file contains the following:

```
20003 Mohtashim M.
20004 Nishant Malik
20005 Amitabh Bachhan
```

When you compile and execute the above program, it produces the following result:

```
20003 Mohtashim M.
20004 Nishant Malik
20005 Amitabh Bachhan
```

Syntax

Following is the syntax to a read record when the file access mode is random:

```
READ file-name RECORD INTO ws-file-structure
   KEY IS rec-key
   INVALID KEY DISPLAY 'Invalid Key'
   NOT INVALID KEY DISPLAY 'Record Details: ' ws-file-structure
END-READ.
```

Example: The following example reads an existing file using indexed organization. This program can be compiled and executed using **JCL** on Mainframes where it will display all the records present in the file. On Mainframes server we do not use text files; instead we use PS files.

Let's assume that the file present on Mainframes have same content as input.txt file in the above example.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

ENVIRONMENT DIVISION.
   INPUT-OUTPUT SECTION.
   FILE-CONTROL.
   SELECT STUDENT ASSIGN TO IN1
      ORGANIZATION IS INDEXED
      ACCESS IS RANDOM
      RECORD KEY IS STUDENT-ID
      FILE STATUS IS FS.

DATA DIVISION.
   FILE SECTION.
   FD STUDENT.
      01 STUDENT-FILE.
      05 STUDENT-ID PIC 9(5).
      05 NAME PIC A(25).

   WORKING-STORAGE SECTION.
   01 WS-STUDENT.
      05 WS-STUDENT-ID PIC 9(5).
      05 WS-NAME PIC A(25).

PROCEDURE DIVISION.
   OPEN INPUT STUDENT.
      MOVE 20005 TO STUDENT-ID.

      READ STUDENT RECORD INTO WS-STUDENT-FILE
         KEY IS STUDENT-ID
         INVALID KEY DISPLAY 'Invalid Key'
         NOT INVALID KEY DISPLAY WS-STUDENT-FILE
      END-READ.

   CLOSE STUDENT.
STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
//IN1 DD DSN=STUDENT-FILE-NAME,DISP=SHR
```

When you compile and execute the above program, it produces the following result:

```
20005 Amitabh Bachhan
```

Write Verb

Write verb is used to insert records in a file. Once the record is written, it is no longer available in the record buffer. Before inserting records into the file, move values into the record buffer and then perform write verb.

Write statement can be used with **FROM** option to directly write records from the working storage variables. From is an optional clause. If the access mode is sequential, then to write a record,s the file must open in Output mode or Extend mode. If the access mode is random or dynamic, then to write a record, the file must open in Output mode or I-O mode.

Syntax

Following is the syntax to read record when the file organization is sequential:

```
WRITE record-buffer [FROM ws-file-structure]
END-WRITE.
```

Following is the syntax to read a record when the file organization is indexed or relative:

```
WRITE record-buffer [FROM ws-file-structure]
   INVALID KEY DISPLAY 'Invalid Key'
   NOT INVALID KEY DISPLAY 'Record Inserted'
END-WRITE.
```

Example: The following example shows how to insert a new record in a new file when the organization is sequential.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

ENVIRONMENT DIVISION.
   INPUT-OUTPUT SECTION.
   FILE-CONTROL.
   SELECT STUDENT ASSIGN TO OUT1
      ORGANIZATION IS SEQUENTIAL
      ACCESS IS SEQUENTIAL
      FILE STATUS IS FS.

DATA DIVISION.
   FILE SECTION.
   FD STUDENT
   01 STUDENT-FILE.
      05 STUDENT-ID PIC 9(5).
      05 NAME PIC A(25).
      05 CLASS PIC X(3).

   WORKING-STORAGE SECTION.
   01 WS-STUDENT.
      05 WS-STUDENT-ID PIC 9(5).
      05 WS-NAME PIC A(25).
      05 WS-CLASS PIC X(3).

PROCEDURE DIVISION.
   OPEN EXTEND STUDENT.
      MOVE 1000 TO STUDENT-ID.
      MOVE 'Tim' TO NAME.
      MOVE '10' TO CLASS.
      WRITE STUDENT-FILE
      END-WRITE.
   CLOSE STUDENT.
STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
//OUT1 DD DSN=OUTPUT-FILE-NAME,DISP=(NEW,CATALOG,DELETE)
```

When you compile and execute the above program, it will add a new record to the output file.

```
1000 Tim         10
```

Rewrite Verb

Rewrite verb is used to update the records. File should be opened in I-O mode for rewrite operations. It can be used only after a successful Read operation. Rewrite verb overwrites the last record read.

Syntax

Following is the syntax to read record when the file organization is sequential:

```
REWRITE record-buffer [FROM ws-file-structure]
END-REWRITE.
```

Following is the syntax to read a record when the file organization is indexed or relative:

```
REWRITE record-buffer [FROM ws-file-structure]
   INVALID KEY DISPLAY 'Invalid Key'
   NOT INVALID KEY DISPLAY 'Record Updated'
END-REWRITE.
```

Example: The following example shows how to update an existing record which we have inserted in previous Write step:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

ENVIRONMENT DIVISION.
   INPUT-OUTPUT SECTION.
   FILE-CONTROL.
   SELECT STUDENT ASSIGN TO IN1
      ORGANIZATION IS INDEXED
      ACCESS IS RANDOM
      RECORD KEY IS STUDENT-ID
      FILE STATUS IS FS.

DATA DIVISION.
   FILE SECTION.
   FD STUDENT
   01 STUDENT-FILE.
      05 STUDENT-ID PIC 9(4).
      05 NAME PIC A(12).
      05 CLASS PIC X(3).

   WORKING-STORAGE SECTION.
   01 WS-STUDENT.
      05 WS-STUDENT-ID PIC 9(5).
      05 WS-NAME PIC A(25).
      05 WS-CLASS PIC X(3).

PROCEDURE DIVISION.
   OPEN I-O STUDENT.
   MOVE '1000' TO STUDENT-ID.

   READ STUDENT
      KEY IS STUDENT-ID
      INVALID KEY DISPLAY ‘KEY IS NOT EXISTING’
   END-READ.

   MOVE 'Tim Dumais' TO NAME.
   REWRITE STUDENT-FILE
   END-REWRITE.
   CLOSE STUDENT.
STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
//IN1 DD DSN=OUTPUT-FILE-NAME,DISP=SHR
```

When you compile and execute the above program, it will update the record:

```
1000 Tim Dumais  10
```

Delete Verb

Delete verb can be performed only on indexed and relative files. The file must be opened in I-O mode. In sequential file organization, records cannot be deleted. The record last read by the Read statement is deleted in case of sequential access mode. In random access mode, specify the record key and then perform the Delete operation.

Syntax

Following is the syntax to delete a record:

```
DELETE file-name RECORD
   INVALID KEY DISPLAY 'Invalid Key'
   NOT INVALID KEY DISPLAY 'Record Deleted'
END-DELETE.
```

Example to delete an existing record:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

ENVIRONMENT DIVISION.
   INPUT-OUTPUT SECTION.
   FILE-CONTROL.
   SELECT STUDENT ASSIGN TO OUT1
      ORGANIZATION IS INDEXED
      ACCESS IS RANDOM
      RECORD KEY IS STUDENT-ID
      FILE STATUS IS FS.

DATA DIVISION.
   FILE SECTION.
   FD STUDENT
   01 STUDENT-FILE.
      05 STUDENT-ID PIC 9(4).
      05 NAME PIC A(12).
      05 CLASS PIC X(3).
   WORKING-STORAGE SECTION.
   01 WS-STUDENT.
      05 WS-STUDENT-ID PIC 9(5).
      05 WS-NAME PIC A(25).
      05 WS-CLASS PIC X(3).

PROCEDURE DIVISION.
   OPEN I-O STUDENT.
   MOVE '1000' TO STUDENT-ID.

   DELETE STUDENT RECORD
      INVALID KEY DISPLAY 'Invalid Key'
      NOT INVALID KEY DISPLAY 'Record Deleted'
   END-DELETE.

   CLOSE STUDENT.
STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
//OUT1 DD DSN=OUTPUT-FILE-NAME,DISP=SHR
```

When you compile and execute the above program, it produces the following result:

```
Record Deleted
```

Start Verb

Start verb can be performed only on indexed and relative files. It is used to place the file pointer at a specific record. The access mode must be sequential or dynamic. File must be opened in I-O or Input mode.

Syntax

Following is the syntax to place the pointer at a specific record:

```
START file-name KEY IS [=, >, <, NOT, <= or >=] rec-key
   INVALID KEY DISPLAY 'Invalid Key'
   NOT INVALID KEY DISPLAY 'File Pointer Updated'
END-START.
```

Close Verb

Close verb is used to close a file. After performing Close operation the variables in the file structure will not be available for processing. The link between program and file is lost.

Syntax

Following is the syntax to close a file:

```
CLOSE file-name.
```