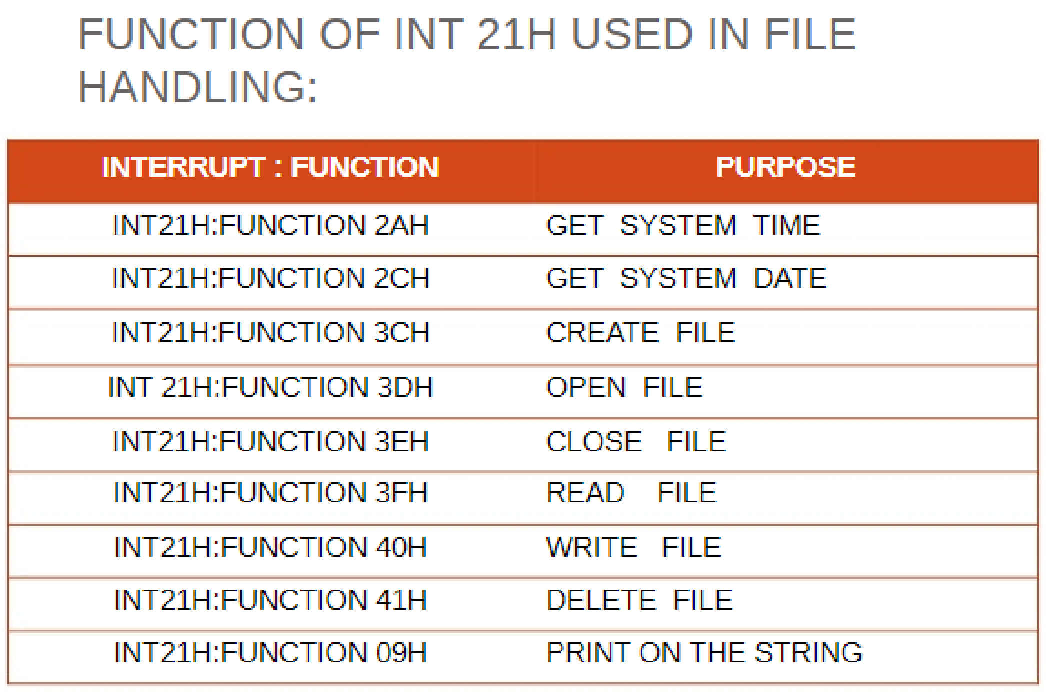
Lab08 – File Handling:

Objective:

- Opening File
- Reading File
- Detecting Next line in File
- Counting Characters in File
- Writing to File
- Appending File
- Closing File



FUNCTION OF INT 21H USED IN FILE HANDLING:

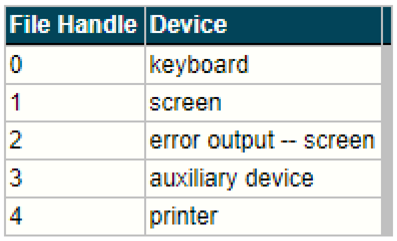
INTERRUPT : FUNCTION	PURPOSE
INT21H:FUNCTION 2AH	GET SYSTEM TIME
INT21H:FUNCTION 2CH	GET SYSTEM DATE
INT21H:FUNCTION 3CH	CREATE FILE
INT 21H:FUNCTION 3DH	OPEN FILE
INT21H:FUNCTION 3EH	CLOSE FILE
INT21H:FUNCTION 3FH	READ FILE
INT21H:FUNCTION 40H	WRITE FILE
INT21H:FUNCTION 41H	DELETE FILE
INT21H:FUNCTION 09H	PRINT ON THE STRING

File Opening:”

- Under data directive, define a string which contains file name and append 0 at end
  - file db “myfile.txt”,0
- Define a buffer in which file contents will be stored after reading
  - buffer db 5000 dup("$")

File Handle:

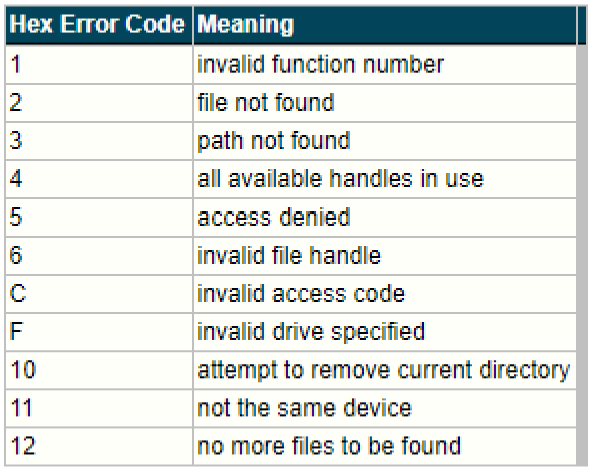
When a file is opened or created in a program, DOS assigns it a unique number called the *file handle*. This number is used to identify the file, so the program must save it.



File Handle	Device
0	keyboard
1	screen
2	error output -- screen
3	auxiliary device
4	printer

File Errors

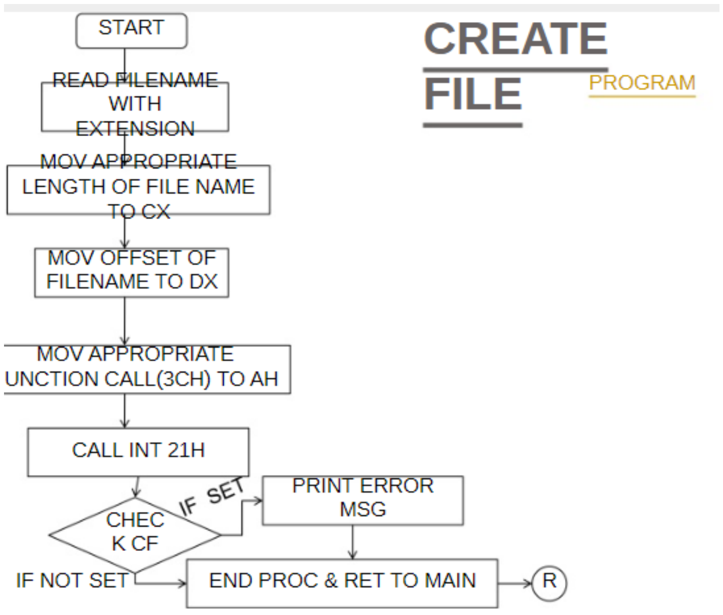
There are many opportunities for errors in INT 21h file handling. DOS identifies each error by a code number. In the functions we describe here, if an error occurs then the CF is set and the code number appears in AX.



Hex Error Code	Meaning
1	invalid function number
2	file not found
3	path not found
4	all available handles in use
5	access denied
6	invalid file handle
C	invalid access code
F	invalid drive specified
10	attempt to remove current directory
11	not the same device
12	no more files to be found

Opening a New File

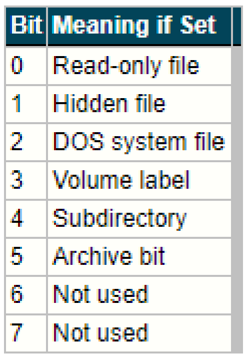
Before a file can be used, it must be opened. To create a new file or rewrite an existing file, the user provides a filename and an attribute and DOS returns a file handle.


CREATE
FILE
PROGRAM
START
READ FILENAME WITH EXTENSION
MOV APPROPRIATE LENGTH OF FILE NAME TO CX
MOV OFFSET OF FILENAME TO DX
MOV APPROPRIATE UNCTION CALL(3CH) TO AH
CALL INT 21H
CHEC K CF
IF SET
PRINT ERROR MSG
IF NOT SET
END PROC & RET TO MAIN
R

Before a file can be used, it must be opened. To create a new file or rewrite an existing file, the user provides a filename and an attribute and DOS returns a file handle.

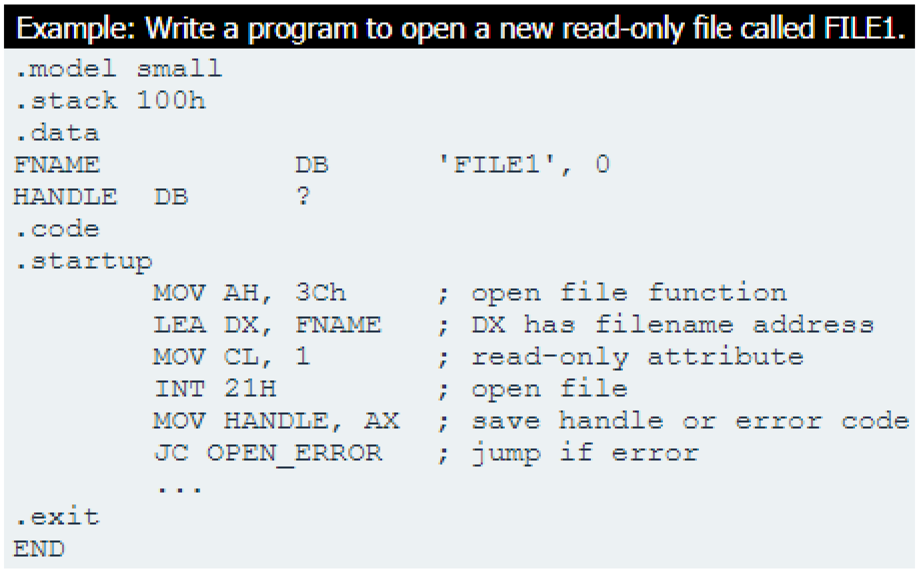
Open a New File/Rewrite a File: INT 21h, function 3Ch

```
Input: AH = 3Ch
       DX = address of file name, which is an ASCII string
                (a string ending with a 0 byte)
       CL = attribute
Output:        if successful, AX = file handle
       Error if CF = 1, error code in AX (3, 4, or 5)
```



Bit	Meaning if Set
0	Read-only file
1	Hidden file
2	DOS system file
3	Volume label
4	Subdirectory
5	Archive bit
6	Not used
7	Not used

Possible errors for this function are 3 (path does not exist), 4 (all file handles in use), or 5 (access denied, which means either that the directory is full or the file is read-only file).



Example: Write a program to open a new read-only file called FILE1.

```
.model small
.stack 100h
.data
FNAME           DB      'FILE1', 0
HANDLE  DB      ?
.code
.startup
        MOV AH, 3Ch     ; open file function
        LEA DX, FNAME   ; DX has filename address
        MOV CL, 1       ; read-only attribute
        INT 21H         ; open file
        MOV HANDLE, AX  ; save handle or error code
        JC OPEN_ERROR   ; jump if error
        ...
.exit
END
```

Opening an Existing File

To open an existing file, there is another function:

```
Open an Existing File: INT 21h, function 3Dh
Input:  AH = 3Dh
        DS:DX = address of file name, which is an ASCII string
                 (a string ending with a 0 byte)
        AL = access code:      0 means open for reading
                               1 means open for writing
                               2 means open for both
Output: if successful, AX = file handle
        Error if CF = 1, error code in AX (2, 4, 5 or 12)
```

Loading File Handler:

- First step is to Load File handler
    - File handler acts as a pointer to file

- mov dx, offset file ; Load address of String “file”
- mov al, 0   ; Open file (read-only)
- mov ah, 3dh   ; Load File Handler and store in ax
- int 21h
- Reading file:
- mov bx,ax   ; Move file Handler to bx
- mov dx, offset buffer   ; Load address of string in which file contents will be stored after reading

- mov ah, 3fh ; Interrupt to read file
- int 21h ; Read file and store the contents in string whose address is stored in dx

Printing Contents of File:

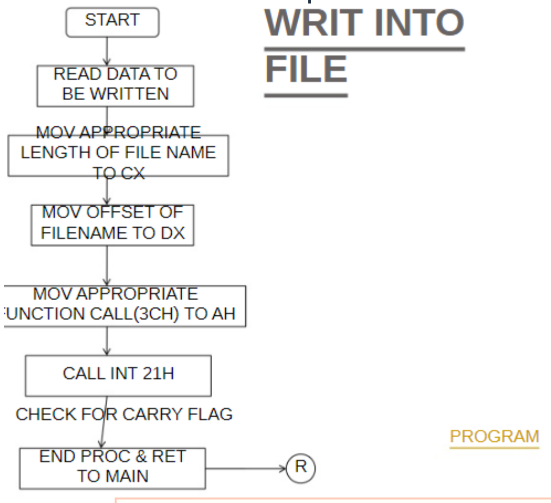
- All the contents of file are stored in string “Buffer”.
- Now print this String using int21h/09h interrupt.

Detecting Next Line in File:

- Contents of file is stored in buffer string.
- Next line will be indicated by “0DH” and “0AH”.
- 46h, 41h, 53h, 54h, 0Dh, 0Ah

Writing to file:

Function 40h writes a specified number of bytes to a file.



START
READ DATA TO BE WRITTEN
MOV APPROPRIATE LENGTH OF FILE NAME TO CX
MOV OFFSET OF FILENAME TO DX
MOV APPROPRIATE UNCTION CALL(3CH) TO AH
CALL INT 21H
CHECK FOR CARRY FLAG
END PROC & RET TO MAIN
R
WRIT INTO FILE
PROGRAM

```
Write to a File: INT 21h, function 40h
Input:  AH = 40h
        BX = file handle
        CX = number of bytes to write
        DS:DX = data address
Output: AX = number of bytes written
        if AX=0 or AX < CX, error (full disk)
        Error if CF = 1, error code in AX (5, 6)
```

Function 40h writes data to a file, but it can also be used to send data to the screen or printer (handles 1 and 4 respectively).

Example: Use function 40h to display a message on the screen.

```
.data
       MSG DB 'Display This Message'
       ...
.code
       ...
       MOV AX, 40h     ; write file function
       MOV BX, 1       ; screen file handle
       MOV CX, 20      ; length of message
       LEA DX, MSG     ; get address of MSG
       INT 21H         ; display MSG
       ...
```

- First step is to Load File handler
    - mov dx, offset file ; Load address of String “file”
    - mov al, 2 ; Open file (read/write)
    - mov ah, 3dh ; Load File Handler and store in ax
    - int 21h
- mov cx, 10 ; Number of bytes to write
- mov bx,ax ; Move file Handler to bx
- mov dx, offset msg ; Load offset of string which is to be written to file
- mov ah, 40h ; Write to file
- int 21h

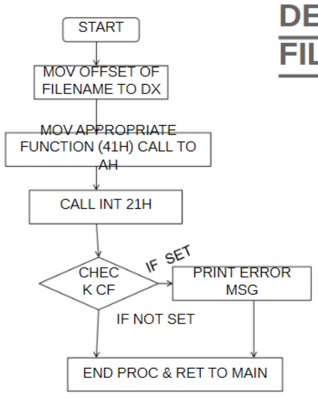
Appending File:

- Move file pointer to end of file before writing to file
- mov cx,0
- mov ah, 42h ; Move file pointer
- mov al, 02h ; End of File
- int 21h

Closing file:

- mov ah, 3eh
- int 21h ;close the file

After a file is processed, it should be closed. This frees the file handle for use with another file.



DELETE FILE

PROGRAM

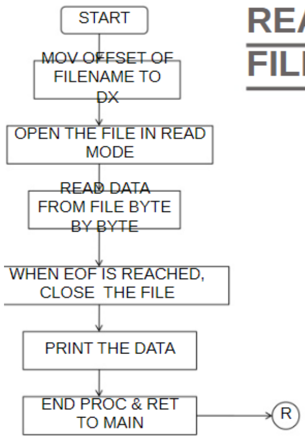
```
Close a File: INT 21h, function 3Eh
Input:  BX = file handle
Output: if CF = 1, error code in AX (6)
```

Example: Write some code to close a file whose handle is stored in variable HANDLE.

```
        MOV AX, 3Eh      ; close file function
        MOV BX, HANDLE   ; get handle
        INT 21H          ; close file
        JC CLOSE_ERROR   ; jump if error
        ...
```

Reading from a File

The following function reads a specified number of bytes from a file and stores them in memory.


START
MOV OFFSET OF FILENAME TO DX
OPEN THE FILE IN READ MODE
READ DATA FROM FILE BYTE BY BYTE
WHEN EOF IS REACHED, CLOSE THE FILE
PRINT THE DATA
END PROC & RET TO MAIN
R

READ FROM FILE

PROGRAM

```
Read from a File: INT 21h, function 3Fh
Input:  AH = 3Fh
        BX = file handle
        CX = number of bytes to read
        DS:DX = memory buffer address
Output: AX = number of bytes actually read
        if AX=0 or AX < CX, end of file encountered
```

Error if CF = 1, error code in AX (5, 6)

Example: Write some code to read a 512-byte from a file. Assume file handle is stored in variable HANDLE, and BUFFER is a 512 byte buffer.

```
.data
        HANDLE DW       ?
        BUFFER DB 512 DUP(0)
        ...
.code
        ...
        MOV AX, 3Fh     ; read file function
        MOV BX, HANDLE  ; get handle
        MOV CX, 512     ; read 512 bytes
        INT 21H         ; read file, AX = bytes read
        JC READ_ERROR   ; jump if error
        ...
```

Activities:

- Write assembly program, which opens a file and print its contents on CONSOLE and closes the file.
- Write Assembly program which writes a String to a file. Define String in program.
- Write Assembly program which appends a String to a file.
- Write Assembly program which takes 5 numbers input from user and write to file.