File Operations

Lab Manual 6

LAB 6 – DOS INTERRUPTS – FILE ACCESS

6.1 Introduction to DOS Interrupts for Files

All DOS files are sequential files. All sequential files are stored and accessed from the beginning of the file towards the end. Files are usually accessed through DOS INT 21H function calls. In this session we will practice, how to create, read and write a file. There are two ways of handling a file. First via the file control block and second via the file handle. We will use file handle, as this is the most common and easier of the two methods.

6.1 File Handle

DOS File Handle Functions - a group of INT 21h . File Handle Functions also permit users to specify file path names, read, write, append file contents etc.. For the purposes of the following discussion, reading means copying all or part of an existing file into memory; writing a file means copying data from memory to a file; rewriting a file means replacing a file's content with other data.

File Handle

When a file is created or opened 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 that information. (Note: a file stream in C/C++/Java is the same thing as a file handle in Assembler.)

There are 5 predefined file handles:

0 keyboard

1 screen

2 error output - usually the screen

3 COM1

4 printer

Additional user-defined files are assigned file handles starting with 5, 6, 7, etc - which are assigned to your files.

File Errors

There can be many types of errors in using INT 21h file handling functions. Each error type is identified by a code number. In the following functions, if an error occurs the CF is set and the code number is stored in AX.

A list of common file-handling errors:

Error Code:	Meaning:
1	invalid function number
2	file not found
3	path not found
4	all available handles in use
5	access denied (file may already be open by another process)
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

THE FILE POINTER

The file pointer is used to locate a position in a file.

When the file is opened, the file pointer is positioned at the beginning of the file.

After a read operation, the file pointer indicates the next byte to be read; after writing a new file, the file pointer is at EOF.

6.2 File Interrupts

Purpose: Create a File – Function $3C_H$ – Create a File

Input:

AH = 3Ch

DS:DX = address of filename (an ASCIIZ string ending with a zero byte)

e.g of ASCIIZ – 'C:\MASM611\BIN\abc.txt',0

CL = attribute

Attribute – Bit map

Bits	7	6	5	4	3	2	1	0
Description	shareable	-	archive	directory	vol. label	system	hidden	read-only

Output:

If successful CF =0 , AX = file handle

Error: if CF = 1, AX = error code (3, 4, or 5)

Possible errors:

1. path doesn't exist
2. all file handles in use
3. access denied --- directory is full or file is read-only

Example: Program Segment open a new read-only file called "FILE1"

```
FNAME     DB     'FILE1.txt', 0
HANDLE    DW     ?
.CODE
          MOV    AH, 3CH          ; open file function
          LEA    DX, FNAME        ; copy address to DX
          MOV    CL, 1            ; read-only attribute
          INT    21H              ; open the file
          MOV    HANDLE, AX       ; handle or err code
```

The above code will create a file FILE1.txt in the bin directory of MASM

6.3 Purpose: Opening a existing File – Function $3D_H$

Input:

AH = $3D_H$

AL = access and sharing modes

0 = open for reading

1 = open for writing

2 = open for read/write

DS:DX = ASCIZ filename

Output:

CF clear if successful, AX = file handle

CF set on error AX = error code (01h,02h,03h,04h,05h,0Ch,56h)

6.4. Purpose: Writing to a File – Function

40_H Input:

AH = 40h

BX = file handle

CX = number of bytes to write

DS:DX = data address

Output:

AX = count of bytes written.

If AX < CX, error (disk full).

If CF = 1, AX = error code (5, 6)

If CX is zero, no data is written, and the file is truncated or extended to the current position

Data is written beginning at the current file position, and the file position is updated after a successful write

The usual cause for AX < CX on return is a full disk

6.5. Purpose: Closing a File – Function $3E_H$

A file should be closed after it has been processed. This frees the file handle for use with another file. If the file is being written, closing the file causes any data remaining in memory to be written to the file, and the file's time, date, and size will be updated in the directory entry.

Input:

AH = 3Eh

BX = file handle

Output:

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

Example: Write a program to create a file called f8.asm and write 'Hello World' into the file.

```
.model tiny
.data
fname1 db 'f8.txt',0
dat1 db "Hello world"
.code
.startup
        mov ah,3ch ; create file
        lea dx, fname1
        mov cl,20h
        int 21h
        mov bx,ax ; file handle is copied
        mov ah,40h
        mov cx,10
        lea dx, dat1
        int 21h
        mov ah,3eh ; close the file
        int 21h
.exit
```

Note: If a file is already created.

6.6.Purpose: Reading File – Function $3F_H$

Input:

AH = 3Fh
BX = file handle
CX = number of bytes to read
DS:DX = memory buffer address

Output:

AX = count of bytes actually read.
If AX = 0 or AX < CX, EOF
If CF = 1, AX = error code (5, 6)

6.7.Purpose: Moving File Pointer 42_H

Input:

AH = 42h

AL = movement code:

0 = relative to beginning of file

1 = relative to current file position

2 = move relative to end of file

BX = file handle

CX: DX = number bytes to move (signed)

Output:

DX:AX = new location in bytes from the beginning of the file

If CF = 1, AX = error code (1, 6)

Note:

CX:DX holds the number of bytes to move the pointer

< 0 => move pointer backward

> 0 => move pointer forward

If CX: DX is too large, the pointer could be moved past the beginning or end of the file. This is not an error, but will cause an error when the next read or write is executed.

If AL = 0 => move pointer from beginning of file (forward)

If AL = 1 => move pointer from current position (forward or backward)

If AL = 2 => move pointer from end of file (backward

6.8.Purpose: Deleting a File 41H

Input

AH = 41H

DS: DX = address of the ASCII-Z string file name

Output

AX = error code if carry is set

6.9. Purpose: Renaming a File

56H Input

AH = 56_H

DS:DX = ASCIZ filename of existing file

ES:DI = ASCIZ new filename

CL = attribute mask

Output

CF clear if successful

CF set on error, AX= error code (02h,03h,05h,11h)

Few More Interesting File Functions [optional features for understanding]

GET/ SET FILE'S LAST-WRITTEN DATE AND TIME

Input

AH = 57_H with AL =01h (Set) AL = 00 (Get)

BX = file handle

Input/ Output

CX = time & DX = date

Time	15-11	10-5	4-0	Day	15-9	8-5	4-0
Format	Hrs	Minutes	Seconds	Format	Year	Month	Day

Output

CF clear if successful

CF set on error AX = error code (01h, 06h)

GET/ SET FILE'S ATTRIBUTE

Input

AH = 43_H

AL =01h (Set) AL = 00 (Get)

DS:DX – ASCIIZ file name

Input/ Output

CX = Attribute

Output

CF clear if successful

CF set on error AX = error code (01h,06h)

Tasks

1. Create a new file of any name and in that file write your name and ID. No. twice on two different lines. Hint: make use of carriage return and next line ASCII codes. Be sure to close the file.
2. To the file resulting from Task1, append your hostel name and room number. Close the file in the end.
3. Open the file you created in Task 2 and read its contents. You can count the number of bytes to be read beforehand. After reading the file, display the data on console using dos function call with ah=09h studied in lab 4. Close the file in the end.
4. Open the file you created in Task 2. Read the entire file one byte at a time and stop after you reach the end of file. After reading the file, display the data using dos function call with ah=09h studied in lab 4. Close the file in the end.
5. Create a new file and write your name and ID. No. in it. Use keyboard to input the data. Hint: Use dos function with ah=0Ah for input from keyboard. Also try renaming the file and deleting it, changing its attributes date of creation & time.