Birzeit University

Faculty of Information Technology

Computer Systems Engineering Department

Assembly Lab ENCS 311 EXP. No. 7

I/O Files

1. Objectives:

The purpose of this experiment is

1. To provide an overview of Files.
2. Introduce File creation , writing and reading
3. Introduce creating libraries

2 Introduction

The data are stored on the disk in the form of files. In Dos and windows files can be accessed using INT 21H. There are two types of files structured and unstructured files. An unstructured file typically refers to a file that is simply a collection of bytes with no meanings to particular byte offsets within the file. A text file is an example of an unstructured file (contains ASCII data). In a structured file, individual bytes or groups of bytes are meant to contain defined data items such as characters, integers, strings, floating point numbers etc, with the ordering of the fields always the same for files of that particular type. The PC world contains many examples of structured files; .BMP (bitmap files) and .WAV (sound files) are two examples of structured files.

3. Access Files

3.1 File creation

A file can be created using INT 21h and function 3CH. A file names is stored in ASCII-Z string and may contain the drive and directory path if needed. The cx must contain the attributes of the file (hidden , read-only,…) for example if bit #0 is ‘1’ then the file is read-only , if bit #2 is ‘1’ then the files is hidden. When cx is zero then it is a normal file. After returning form int 21h if the carry is cleared then there is no error occurred and the AX contains a **file handler**. The file handler is number that is used to refer to the file if it created or opened.

3.2 Writing to a file

Before writing to a file it must have been created or opened. Then the file handler is used to refer to the file whenever data are written. Function 40H is used to write to a file in which BX must contain the file handler , cx contains the number of bytes to be written and DS:DX contains the address of the data to be written.

3.3 Opening, Reading, and Closing a file.

To read a file it must first be opened. Function 3DH is used to open the file , AL must contain the operation allowed for the opened file (AL=00 read 01 write , 02 read or write)

Function 3FH causes a file to be read. As with write function BX contains the file handler, CX contains the number of bytes to be read and DS:DX contains the memory location where the data will be stored. As with all disk operations, the carry flag indicates an error with a logic 1. If logic 0 is indicated the AX indicates the number of bytes read from the file.

Closing a file is very important. If a file is left open some serious problems can occur. Function 3E is used to close files.

3.4 File pointer:

When a file is opened, written or read a file pointer addresses the current location in the sequential file. File pointer is 32 bit number that addresses any byte in a file. Once the file is opened, the file pointer can be changed with move file pointer function number 42h. a file pointer can be moved from start of the file (AL=0), from the current location (AL=01) or from the end of the file (AL=02). The distance moved by the file pointer is spicifed by CX (most significant) and DX (least significant part). BX must contain the file handler

4. Creating and Using Libraries

The following steps show how to create library named “mylib”:

- Store the code in mylib.asm:
- Assemble the library:

```
tasm mylib.asm
```

- Create the library form the object file using “**TLIB”** utility:

TLIP libname +libobjectfile

TLIP mylib +mylib.obj

Now you can use the library. For example the file uselib.asm (Program 3) will use the function writestring from the library mylib (Program 2)

Assemble:

tasm uselib.asm

Linking:

tlink uselib.obj mylib.lib

uselib.exe

5. Pre Lab Work:

1. Study program 1, and explain how it works?
2. Write, assemble and link program 1. You will run it in the lab using TASM
3. Modify the program such that it reads from the file you created and write the data it has been read on the screen
4. Assemble mylib.asm and create a library then use it (uselib.asm)
5. Bring your work to the lab

6. Lab Work:

1. Write, assemble and link program 4. Study how it does work!
2. Write a library for files operations, it should contain the functions, createfile, openfile, readfile , writefile, closefile. Then use this library to write a program the reads data from a file (FILENAME.IN) provided by the user then store the data REVERSED on a file has the same name with extension OUT.
3. Repeat step 2 but append the reversed data to the same file.

Program 1

```
;THIS PROGRAM CREATE AND WRITE THEN CLOSE A FILE
Title “PROGRAM71”
.MODEL SMALL
.STACK 100
.DATA
FILENAME DB "DATA.TXT",0
TEXT DB "WELCOME 311",0
FHAND DW ?
.CODE
MOV AX,@DATA
MOV DS,AX

MOV AH,3CH ; CREATE A FILE
MOV CX,0 ; NORMAL ATTRIBUTES
LEA DX,FILENAME ; THE ADDRESS OF FILE NAME SHOULD BE IN DX
INT 21H
MOV FHAND,AX  ;FILE HANDLE IS RETURNED IN AX , STORE IT WE NEED IT LATER


MOV AH,40H ;  WRITE TO FILE
MOV BX,FHAND ; THE FILE HANDLE
MOV CX,12 ; NUMBER OF BYTES TO BE WRITTEN
LEA DX,TEXT ; THE ADDRESS OF DATA TO BE WRITTEN SHOULD BE IN DX
INT 21H
MOV AH,3EH
INT 21H

MOV AH,4CH
INT 21H
END
```

Program 2

```
;PROGRAM MYLIB.ASM
        PUBLIC WRITESTRING
        .MODEL  SMALL
        .STACK 100H
        .CODE
        WRITESTRING         PROC
        MOV AH,9
        INT 21H
        RET
        WRITESTRING         ENDP
        END
```

Program 3

```
        ;PROGRAM USELIB.ASM
        .MODEL  SMALL
        .STACK 100H
        .DATA
        RSTRING DB "THE RESULT IS: ","$"
        .CODE
        ; WRITESTRING PROCEDURE FOUND IN 'MYLIB.LIB'

                EXTRN WRITESTRING:PROC

                MOV     AX,@DATA
                MOV   DS,AX

                LEA DX,RSTRING
                CALL WRITESTRING

        MOV AH,4CH
        INT 21H
        END
```

Program 4

```
;THIS PROGRAM READ FROM A FILE UNTIL THE END
TITLE “PROGRAM72”
.MODEL SMALL
.STACK 100
.DATA
FILENAME DB "DATA.TXT",0
TEXT DB 250 DUP(?)
FHAND DW ?
.CODE
MOV AX,@DATA
MOV DS,AX

MOV AH,3DH ; READ A FILE
MOV AL,02

LEA DX,FILENAME ; THE ADDRESS OF FILE NAME SHOULD BE IN DX
INT 21H
MOV FHAND,AX    ;FILE HANDLE IS RETURNED IN AX

 MOV SI,0
L:
MOV AH,3FH                  ; THE FILE HANDLE
MOV CX,1                   ; NUMBER OF BYTES TO BE READ
LEA DX,TEXT+SI             ; THE ADDRESS OF DATA TO BE READ
INT 21H

CMP AX,0
JE EXIT
INC SI
JMP L

EXIT:
MOV BYTE PTR TEXT+SI,"$"
MOV AH,3EH
INT 21H

MOV AH,9
MOV DX, OFFSET TEXT
INT 21H

MOV AH,4CH
INT 21H
END
```