
Lecture 10: File Processing

Dragan Mirkovic
Department of Computer Science
University of Houston

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Announcements

- Today we will start with file processing routines
 - Chapter 18 in text

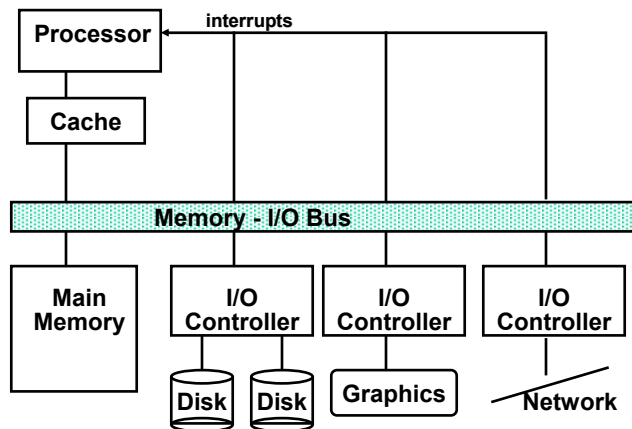
D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Introduction

- I/O is an important part of the information system
- I/O system performance may be limited by mechanical delays (disk I/O)
 - < 10% per year (IO per sec)
- Amdahl's Law: system speed-up limited by the slowest part!
- I/O bottleneck:
 - Diminishing fraction of time in CPU
 - Diminishing value of faster CPUs

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

I/O Systems



D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

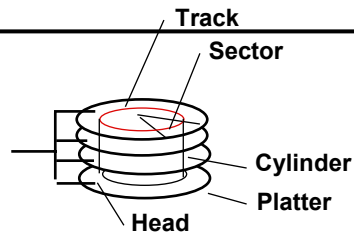
Storage Technology Drivers

- Driven by the prevailing computing paradigm
 - 1950s: migration from batch to on-line processing
 - 1990s: migration to ubiquitous computing
 - computers in phones, books, cars, video cameras, ...
 - nationwide fiber optical network with wireless tails
- Effects on storage industry:
 - Embedded storage
 - smaller, cheaper, more reliable, lower power
 - Data utilities
 - high capacity, hierarchically managed storage

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Devices: Magnetic Disks

- Purpose:
 - Long-term, nonvolatile storage
 - Large, inexpensive, slow level in the storage hierarchy
- Characteristics:
 - Seek Time (~8 ms avg)
 - positional latency
 - rotational latency
- Transfer rate
 - 10-60 MByte/sec
 - Blocks
- Capacity
 - Gigabytes
 - Quadruples every 2 years (aerodynamics)



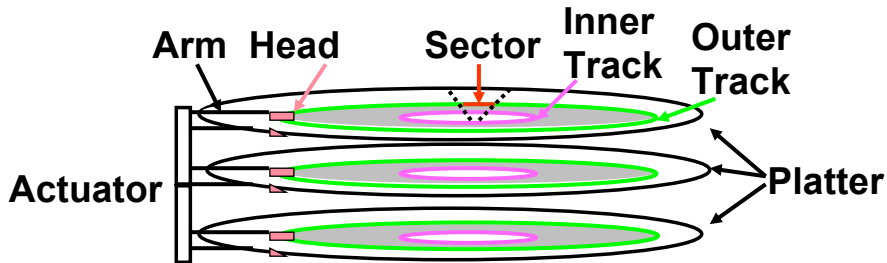
7200 RPM = 120 RPS => 8 ms per rev
ave rot. latency = 4 ms
128 sectors per track => 0.25 ms per sector
1 KB per sector => 16 MB / s

Response time
= Queue + Controller + Seek + Rot + Xfer

Service time

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

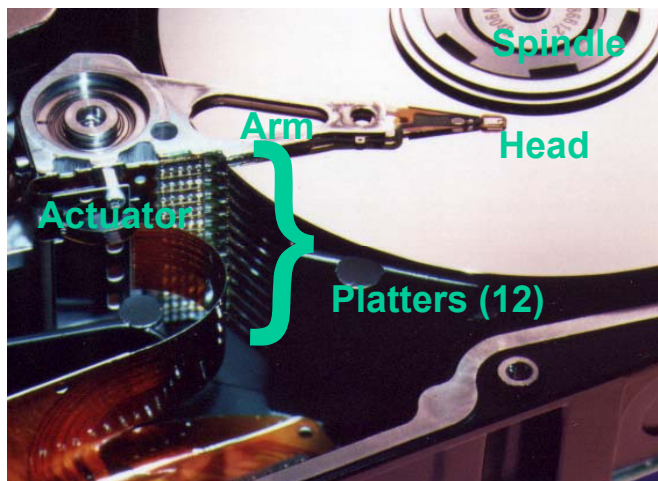
Disk Device Terminology



- Several platters, with information recorded magnetically on both surfaces (usually)
- Bits recorded in tracks, which in turn divided into sectors (e.g., 512 Bytes)
- Actuator moves a head (end of arm, 1/surface) over track ("seek"), select surface, wait for sector rotate under head, then read or write
 - "Cylinder": all tracks under heads

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Photo of Disk Head, Arm, Actuator



D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Disk Storage Fundamentals

- Hardware level – physical layout of the disk
 - Tracks, cylinders, and sectors
 - Sector = 512 byte segment of a track
 - Each sector is assigned a logical sector number
- Software – logical layout
 - Clusters = logical group of adjacent sectors
 - directories and files

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

DOS File System

- Similar to UNIX file system
 - Files arranged in a hierarchical system of directories
 - DOS keeps track of each file by
 - Storing its starting cluster number in the disk directory
 - Storing all remaining cluster numbers in a separate table (FAT)
 - Each file occupies an integer number of clusters
 - In most of the cases the last cluster is only partially used
- FAT is a map of all clusters on the disk

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Directory Structure

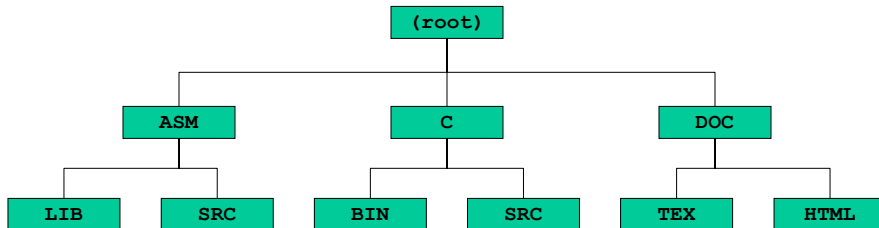
- Every disk has a root directory
 - Primary list of files on the disk
 - May contain the names of other directories
- Example:

```
Directory of C:\
08/27/2003  11:00 AM    <DIR>        apps
04/30/2003  01:58 PM    <DIR>        compaq
05/16/2003  09:19 AM    <DIR>        Documents and Settings
            ...
05/01/2003  05:36 PM                600 winscp.RND
08/21/2003  08:58 AM    <DIR>        WUTemp
                2 File(s)            634 bytes
                15 Dir(s)  904,130,560 bytes free
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Directory Structure...

- The result is a treelike structure with the root at the top
- Each directory and file name is qualified by the names of the directories above it
 - **Path:** C:\ASM\SOURCE\prog1.asm



D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

DOS File Manipulation

- DOS has a variety of file manipulation routines
- Similar to UNIX/C functions
 - Open, close, delete, read write, etc.
- Use of **file handles** (a small integer)
 - Indicates the file in file manipulation routines
 - Device independent system
 - Same programs can be used for a variety of devices
 - Main operations:
 - Sequential read and write
 - Random access by changing the current position

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Standard file handles

- At the start of each program 5 standard file handle are available:

<code>stdin</code>	0	The keyboard
<code>stdout</code>	1	The display
<code>stderr</code>	2	The display
<code>stdaux</code>	3	The standard aux. device. (COM1)
<code>stdprn</code>	4	The standard printer dev. (LPT1)

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

I/O Redirection and piping

- DOS can use the device independence to implement I/O redirection and piping
- Examples:
 - redirection of the standard input and output
`PROG < infile > outfile`
 - Redirection of the standard output to the standard printer
`PROG > PRN`
 - Piping of the standard output of one program to the standard input of another
`dir | more`
 - Redirection and piping together
`dir | more > PRN`

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Basic File Manipulation

- DOS file names are given by 0-terminated variable length character strings (ASCIZ, or ASCII strings)
- Example:

```
fName    DB    'A:\src\myprog.asm', 0
```
- In order to manipulate a file its name must be associated with a file handle.
- One can use one of two DOS functions:
 - The Open File function (3Dh) for existing files
 - The Create File function (3Ch) for new files
- Only a limited number of files can be open at once

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Create File (3Ch)

- Creates a new file
 - The file is automatically opened for input/output
 - CX contains the file attribute bits
 - 00h Normal file
 - 01h Read-only file
 - 02h Hidden file
 - 04h System file
 - DS:DX points to the filename
 - If successful, AX contains a valid file handle
 - If not CF=1 and AX contains the error code

- Example:

```
.data
newfile db 'C:\newf1.doc', 0
handle dw ?
.code
mov ah, 3Ch ; Open file
mov cx, 0 ; File attrib.
mov dx, OFFSET newfile
int 21h
jc display_error
mov handle, ax
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Open File (3Dh)

- Opens an existing file in one of three modes:
 - Input, output, input/output
 - AL contains the file mode
 - 0 Input (read only)
 - 1 Output (write only)
 - 2 Input-output
 - DS:DX points to the filename
 - If successful, AX contains a valid file handle
 - If not CF=1 and AX contains the error code

- Example:

```
.data
fname db 'C:\file1.doc', 0
inHandle dw ?
.code
mov ah, 3Dh ; Open file
mov al, 0 ; File mode
mov dx, OFFSET fname
int 21h
jc display_error
mov inHandle, ax
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Close File (3Eh)

- Used to close an opened file
 - BX contains the file handle
 - This function flushes the internal file buffer and updates the file size, time and date stamps
 - If not successful CF=1 and AX contains the error code

- Example:

```
.data
fname db 'C:\file1.doc', 0
inHandle      dw      ?
.code
mov     ah, 3Eh ; Close file
mov     bx, inHandle
int     21h
jc      display_error
```

Read From File or Device (3Fh)

- After opening a file for input you can use this function to read from the file
 - BX contains the file handle
 - CX contains the number of bytes required
 - DS:DX points to the input buffer
 - On output AX contains the number of bytes read
 - If AX < CX EOF reached before the requested number of bytes was read
 - CF = 1 indicates an error

Example

```
.data
buffsize = 512
inHandle    dw    ?
buffer      db    buffsize dup(0)
.code
mov  ah, 3Fh ; Read from file or device
mov  bx, inHandle
mov  cx, buffsize
mov  dx, OFFSET buffer
int  21h
jc   display_error
cmp  ax, cx          ; compare to bytes requested
jbe  Exit
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Write to File or Device (40h)

- After opening a file for output you can use this function to write to the file
 - BX contains the file handle
 - CX contains the number of bytes to write
 - DS:DX points to the output buffer
 - On output AX contains the number of bytes written
 - If AX < CX : an I/O error has occurred, ex. The disk could be full.
 - CF = 1 indicates an error

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Example

```
.data
buffsize = 512
outHandle    dw    ?
buffer       db    buffsize dup(0)
.code
mov    ah, 40h ; Write to file/device
mov    bx, outHandle
mov    cx, buffsize
mov    dx, OFFSET buffer
int    21h          ; Call DOS
jc     display_error ; error? Display message
cmp    ax, cx       ; all bytes written?
jne    close_file   ; no: disk full
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Random File Access

- Use DOS function 42h (move file pointer)
- The file must already be open
- The input registers are:
 - AH 42h
 - AL Method code (type of offset)
 - 0 offset from the beginning of the file
 - 1 offset from the current location
 - 2 offset from the end of the file
 - BX File handle
 - CX 32-bit offset, high
 - DX 32-bit offset, low

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Example

- We can use the function 42h to append to a file
 - Use method code 2 to position the file pointer to the end of the file

```
SeekEOF  proc
    pusha                ; Save all registers
    mov     ah, 42h      ; Position the file pointer
    mov     al, 2        ; relative to end of file
    mov     bx, handle   ;
    mov     cx, 0        ; offset, high
    mov     dx, 0        ; offset low
    int    21h          ; call DOS
    popa                ; restore registers
    ret
SeekEOF  endp
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Additional I/O routines

- Provide additional functionality, usually beyond that allowed at the command prompt
 - Get/Set File Attribute (43h)
 - Delete File (41h)
 - Rename File (56h)
 - Get/Set File Date/Time (57h)
 - Find First Matching File (4Eh)
 - Find Next Matching File (4Fh)
 - Create or open file (716Ch)
 - Get file creation date and time (5706h)

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Get/Set File Attribute

- Retrieve or change the attribute of a file
- Registers usage:
 - AH 43h
 - AL (0 = get attribute, 1 = set attribute)
 - CX New attribute
 - DS:DX Points to an ASCII string with the file specification
- The carry flag is set if the function fails
- If AL = 0 (get attr.) the file attribute is in CX

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

File Attributes

- Each bit indicates a specific property (8-bits total)
- The meaning of the 1st three bits:

Attribute	Value
Normal file	00
Read-only file	01
Hidden file	02
Hidden, read-only file	03
System file	04
Hidden, read-only, system file	07

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Example

- The following instructions set a file's attributes to hidden and read-only:

```
.data
fname    db    'test.txt',0
.code
mov      ah, 43h
mov      al, 1      ; set file attr.
mov      cx, 3      ; hidden, read-only
mov      dx, offset fname
int      21h
jc       display_error
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Delete file (41h)

- Set DS:DX to the address of an ASCII string containing a file specification
 - It can contain drive and path name
 - No wild characters
- Example:

```
.data
fname    db    'test.txt',0
.code
mov      ah, 41h      ; delete file
mov      dx, offset fname
int      21h
jc       display_error
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Rename (move) file (56h)

- The old name in `DS:DX`
- The new name in `ES:DI`
 - No wild characters allowed
- `CF = 1` indicates an error
- Example:

```
.data
oldname    db    'C:\tmp\test.asm',0
newname    db    'C:\asm\test.asm',0
.code
mov ah, 56h    ; rename file
mov dx, offset oldname
mov di, offset newname
int 21h
jc display_error
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Find First Matching File (4Eh)

- Finds a file that matches a certain specifications
- The file specification is in `DS:DX`
 - It can include wild characters (* and ?)
 - The search is restricted to files with specific attributes
 - Creates file description at current DTA
- Example:

```
.data
fspec      db    'C:\src\asm\*.asm',0
.code
mov        ah, 4Eh ; find file
mov        cx, 0    ; normal files only
mov        dx, offset fspec
int 21h
jc display_error
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Disk Transfer Address (DTA)

- DTA is an area (43 bytes) reserved for the transfer of file data to memory
- It contains the file information
 - 0 – 20 Reserved by DOS
 - 21 Attribute
 - 22 – 23 Time stamp
 - 24 – 25 Date stamp
 - 26 – 29 Size (double word)
 - 30 – 42 File name (ASCIIZ)

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Using the DTA

- The 4Eh (find first matching file) function returns the file description in the default DTA (PSP + 80h)
 - It can be reset to a buffer inside the data segment using the 1Ah function
 - Example:
 - Set the DTA to a buffer DTAbuf
- ```
.data
DTAbuf db 43 dup (?)
 mov ah, 1Ah ; set DTA
 mov dx, offset DTAbuf ; DTA buffer
 int 21h
```

- There is a 4Fh function (find next matching file)

---

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

# File Organization

---

- Many files are organized as a collection of blocks or records
  - Simplifies the file handling
  - Faster search
- The first record/block in a file (header) may have a special purpose
  - Information about the rest of the file:
    - File type, size, number of blocks/records, ...

---

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

# Data Structures in MASM

---

- You should always read header (and other data structures) into an equivalent data structure
- Example: DTA

```
DTAbuf db 43 dup (?)
```

- Is not very convenient

- Breaking up `DTAbuf` is more descriptive:

```
DTAbuf db 21 dup (?) ; DOS reserved field
```

```
DTAattr db ? ; file attribute
```

```
fileTime dw ? ; time stamp
```

```
fileDate dw ? ; date stamp
```

```
fileSize dd ? ; file size
```

```
fileName db 13 dup (0) ; file name
```

---

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

## Data Structures ...

---

- If the same set of possibly dissimilar data types is used more than once one can declare it as a **data structure**
- The structure components can be accessed either as a unit or separately
- A MASM structure is similar to a **struct** in C or a **STRUCTURE** in FORTRAN
- The steps when using structures
  - Declare a structure type
  - Define one or more variables having that type
  - Reference the fields directly or indirectly

---

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

## Declaring Structure Types

---

- Declaration creates a template for data
  - Data sizes and, optionally the initial values
- Syntax:  
*name* **STRUCT** [[*alignment*]] [[[, **NONUNIQUE**]]  
*fielddeclarations*  
*name* **ENDS**
- Declares a structure type having the specified *fielddeclarations*. Each field must be a valid data definition.

---

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

# Defining Structure Variables

---

- Once a structure type is declared you can define variables of this type
  - Allocates the memory in the current segment
- Structure fields can be initialized when the type is declared or when a variable is declared
- Syntax:

```
[name] typename <initializer[,initializer]>
```

  - `name` and `initializers` are optional
  - The angle brackets (<>) are required
  - <> indicates that the default values should be used

---

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

# Example

---

- DTA Structure:

```
DTAblock struct
 DTAbuf db 21 dup (?) ; DOS reserved field
 DTAattr db ? ; file attribute
 fileTime dw ? ; time stamp
 fileDate dw ? ; date stamp
 fileSize dd ? ; file size
 fileName dd 13 dup(0) ; file name
DTAblock ends
```

- The variables of DTAblock type can be specified and initialized as

```
myDTA DTAblock <>
```

---

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

## Referencing Structure Variables

---

- If we define

```
myDTA DTAblock <>
```

- The following expressions will return the same value (43)

```
sizeof myDTA
```

```
sizeof DTAblock
```

- Field references:

```
.data
```

```
myDTA DTAblock <>
```

```
.code
```

```
mov dx, myDTA.fileTime; Gets the time stamp
```

---

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

## Putting it all together

---

- Lets write a simple file manipulation program that will search the current directory for the 1<sup>st</sup> occurrence of the .ASM file and if such a file exist it will print the file name on the screen, or otherwise print an error message.

```
.DATA
myDTA DTAblock <>
 db 13,10,'$' ; append to the file name
allASM db '*.asm', 0 ; search string
msgErr db 'Error: file not found!',13,10,'$'

.CODE
lsASM PROC
 _Begin
 mov ah, 1Ah ; set DTA
 mov dx, offset myDTA ; DTA buffer
 int 21h
```

---

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

# Example

---

```
mov ah, 4Eh ; find file
mov cx, 0 ; normal files only
mov dx, offset allASM ; all .asm files
int 21h
jc display_error ; print error message
mov ah, 9h ; print file name
mov dx, offset myDTA.fileName
int 21h
_Exit 0
display_error:
mov ah, 9h
mov dx, offset msgErr
int 21h
_Exit 1
lsASM ENDP
END lsASM
```

---

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004