# Chapter 4

# File Systems





## Long-term Information Storage

1. Must store large amounts of data
2. Information stored must survive the termination of the process using it
3. Multiple processes must be able to access the information concurrently

# File Naming

8.[3]

האם מערכת הפעלה צריכה
לאכוף את פורמט הקובץ?

מה היתרון בפורמט זה ?

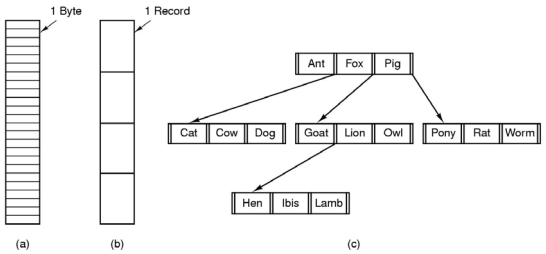
ניידות

| Extension | Meaning |
|---|---|
| file.bak | Backup file |
| file.c | C source program |
| file.gif | Compuserve Graphical Interchange Format image |
| file.hlp | Help file |
| file.htm | World Wide Web HyperText Markup Language document |
| file.jpg | Still picture encoded with the JPEG standard |
| file.mp3 | Music encoded in MPEG layer 3 audio format |
| file.mpg | Movie encoded with the MPEG standard |
| file.o | Object file (compiler output, not yet linked) |
| file.pdf | Portable Document Format file |
| file.ps | PostScript file |
| file.tex | Input for the TEX formatting program |
| file.txt | General text file |
| file.zip | Compressed archive |

Typical file extensions.

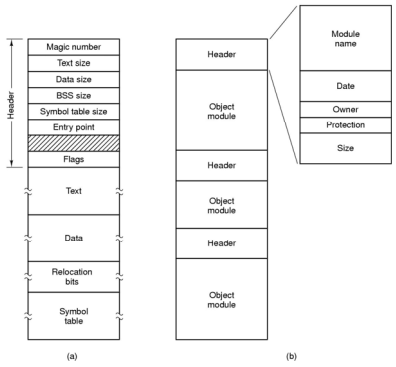


# File Structure




- Three kinds of files
  - byte sequence
  - record sequence
  - tree

# File Types




(a) An executable file (b) An archive



# File Access

- Sequential access
  - read all bytes/records from the beginning
  - cannot jump around, could rewind or back up
  - convenient when medium was mag tape
- Random access
  - bytes/records read in any order
  - essential for data base systems
  - read can be …
    - move file marker (seek), then read or …
    - read and then move file marker

# File Attributes

| Attribute | Meaning |
|---|---|
| Protection | Who can access the file and in what way |
| Password | Password needed to access the file |
| Creator | ID of the person who created the file |
| Owner | Current owner |
| Read-only flag | 0 for read/write; 1 for read only |
| Hidden flag | 0 for normal; 1 for do not display in listings |
| System flag | 0 for normal files; 1 for system file |
| Archive flag | 0 for has been backed up; 1 for needs to be backed up |
| ASCII/binary flag | 0 for ASCII file; 1 for binary file |
| Random access flag | 0 for sequential access only; 1 for random access |
| Temporary flag | 0 for normal; 1 for delete file on process exit |
| Lock flags | 0 for unlocked; nonzero for locked |
| Record length | Number of bytes in a record |
| Key position | Offset of the key within each record |
| Key length | Number of bytes in the key field |
| Creation time | Date and time the file was created |
| Time of last access | Date and time the file was last accessed |
| Time of last change | Date and time the file has last changed |
| Current size | Number of bytes in the file |
| Maximum size | Number of bytes the file may grow to |

איפה נשמור את תכונות הקובץ?
א. בתחילת הקובץ.
ב. בשם הקובץ
ג. בקובץ נוסף
ד. בספרייה

Possible file attributes



# File Operations

1. Create
2. Delete
3. Open
4. Close
5. Read
6. Write
7. Append
8. Seek
9. Get attributes
10. Set Attributes
11. Rename

## An Example Program Using File System Calls (1/2)

```
/* File copy program. Error checking and reporting is minimal. */

#include <sys/types.h>                  /* include necessary header files */
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]);       /* ANSI prototype */

#define BUF_SIZE 4096                   /* use a buffer size of 4096 bytes */
#define OUTPUT_MODE 0700                /* protection bits for output file */

int main(int argc, char *argv[])
{
    int in_fd, out_fd, rd_count, wt_count;
    char buffer[BUF_SIZE];

    if (argc != 3) exit(1);             /* syntax error if argc is not 3 */
```



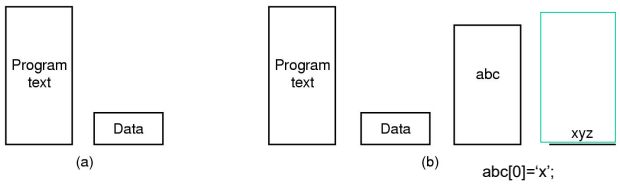
## An Example Program Using File System Calls (2/2)

```
    /* Open the input file and create the output file */
    in_fd = open(argv[1], O_RDONLY);    /* open the source file */
    if (in_fd < 0) exit(2);             /* if it cannot be opened, exit */
    out_fd = creat(argv[2], OUTPUT_MODE);  /* create the destination file */
    if (out_fd < 0) exit(3);            /* if it cannot be created, exit */

    /* Copy loop */
    while (TRUE) {
        rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
    if (rd_count <= 0) break;           /* if end of file or error, exit loop */
        wt_count = write(out_fd, buffer, rd_count); /* write data */
        if (wt_count <= 0) exit(4);     /* wt_count <= 0 is an error */
    }

    /* Close the files */
    close(in_fd);
    close(out_fd);
    if (rd_count == 0)                  /* no error on last read */
        exit(0);
    else
        exit(5);                        /* error on last read */
}
```

# Memory-Mapped Files




(a) Segmented process before mapping files into its address space

(b) Process after mapping existing file *abc* into one segment creating new segment for *xyz*

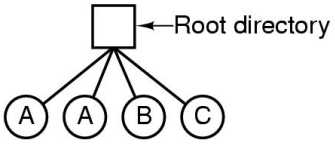
ייתרון: לא דורש seek

חיסרון: 1. אי אפשר לדעת גודל קובץ אלא רק עד כדי דף.
2. סכנת חוסר עקביות בין תהליכים
3. קובץ יכול להיות גדול מ GB4 הכוונה גדול ממרחב הזיכרון הווירטואלי



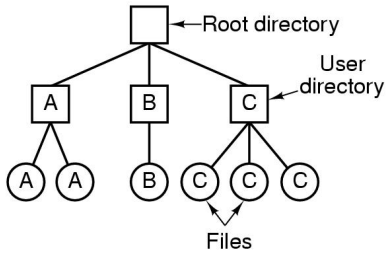
# Directories

## Single-Level Directory Systems




- A single level directory system
  - contains 4 files
  - owned by 3 different people, A, B, and C

# Two-level Directory Systems

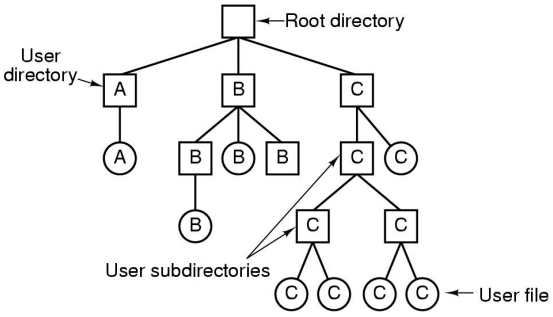



1. פרטיות
2. ספריות משותפות

Letters indicate *owners* of the directories and files



# Hierarchical Directory Systems

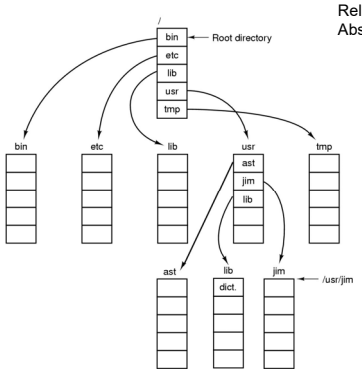



A hierarchical directory system

# Path Names

הדוגמה לא מדוייקת
משום שבפועל מצביע ל
i-node

Relative path ../../
Absolute path




A UNIX directory tree



# Directory Operations

1. Create
2. Delete
3. Opendir
4. Closedir
5. Readdir
6. Rename
7. Link –
   a) symbolic link
   b) Hard link
8. Unlink

# Implementing Files (1)

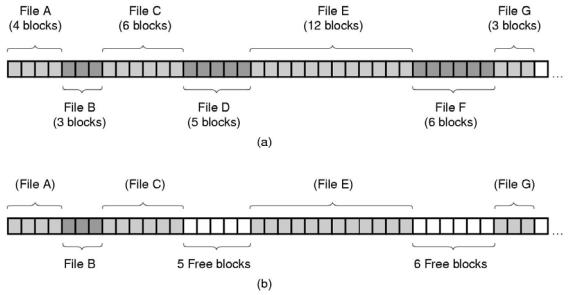
שיטה 1: רציף

ייתרון: הקובץ וגודלו

חסרון: ריסוק חיצוני

ואין אפשרות גדילה

ייתרונות: cdr,dvdr




(a) Contiguous allocation of disk space for 7 files

(b) State of the disk after files *D* and *E* have been removed



# Implementing Files (2)

שיטה 2: בתחילת בלוק יש

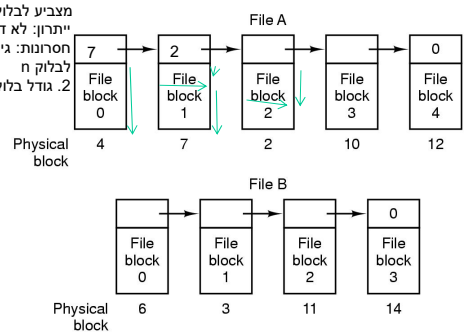
מצביע לבלוק הבא

ייתרון: לא דורש רציפות.

חסרונות: גישה אקראית

לבלוק n

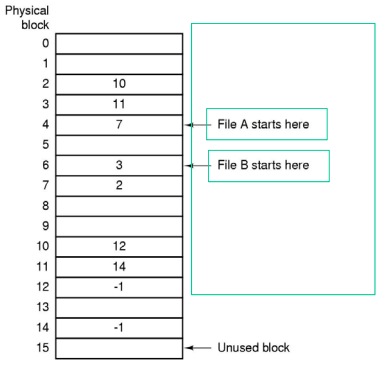
2. גודל בלוק, לא בחזקת 2




Storing a file as a linked list of disk blocks

# Implementing Files (3)

ייתרון: נשמר ב RAM
ולכן גישה אקראית מהירה
חיסרון: נשמר ב ram
והגודל תלוי לינארית בדיסק ולא ב ram



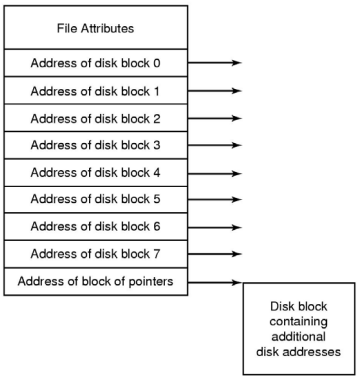

Linked list allocation using a file allocation table in RAM



# Implementing Files (4)

I-node

חיסרון :
1. דורש קובץ נוסף ולכן צריך קריאה נוספת בנפרד
2. דורש עוד בלוקים להחזקת המצביעים לבלוקים




An example i-node

- Symbolic link==soft link ==קיצור דרך

  ייתרון : אפשר לעשות קיצור דרך למערכות מבוזרות ושונות -> URL

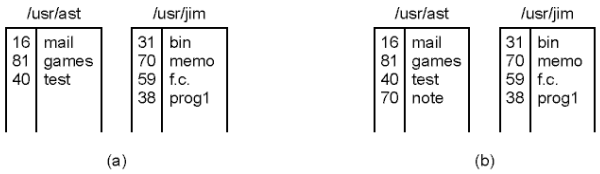
  חיסרון: מחיקת הקובץ תוביל לקישור שבור

- Hard link

  מחיקת המקור לא תמחק את אלה שמקושרים



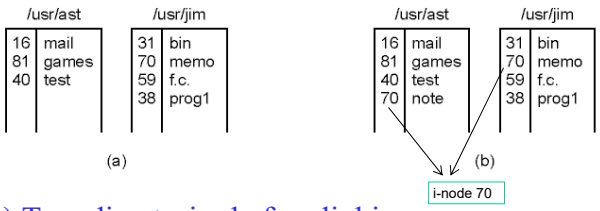
# System Calls (3)




(a) Two directories before linking */usr/jim/memo* to ast's directory

(b) The same directories after linking

# System Calls (3)

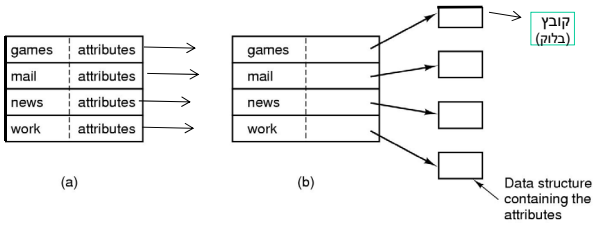



(a) Two directories before linking */usr/jim/memo* to ast's directory

(b) The same directories after linking



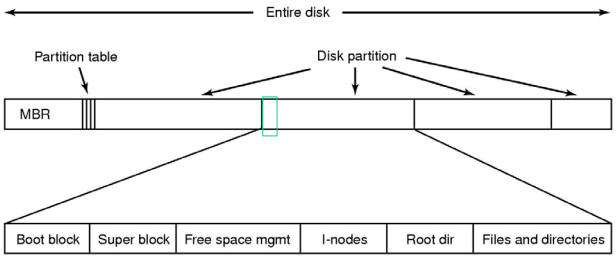
# Implementing Directories (1)




(a) A simple directory
- fixed size entries
- disk addresses and attributes in directory entry

(b) Directory in which each entry just refers to an i-node

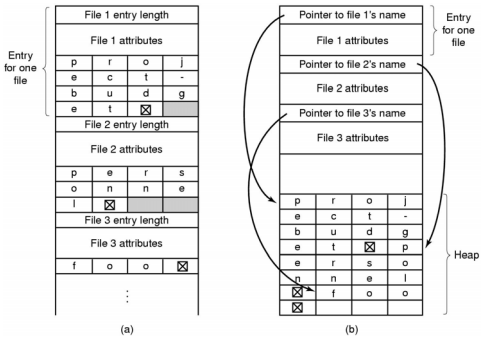
# File System Implementation




A possible file system layout

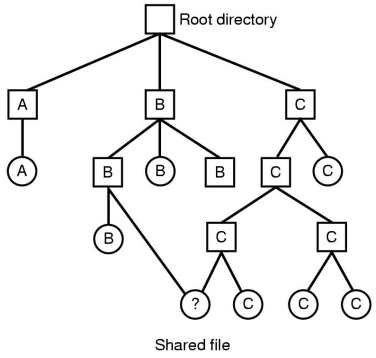


# Implementing Directories (2)




- Two ways of handling long file names in directory
  - (a) In-line
  - (b) In a heap

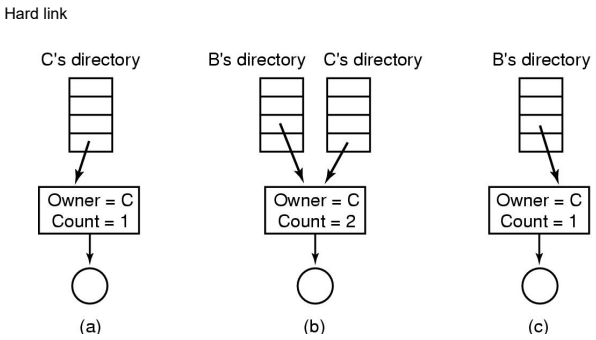
# Shared Files (1)




File system containing a shared file



# Shared Files (2)

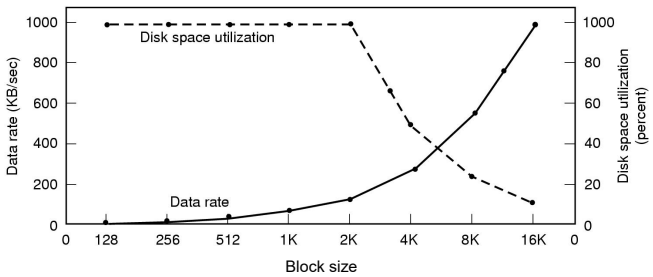



(a) Situation prior to linking

(b) After the link is created

(c)After the original owner removes the file

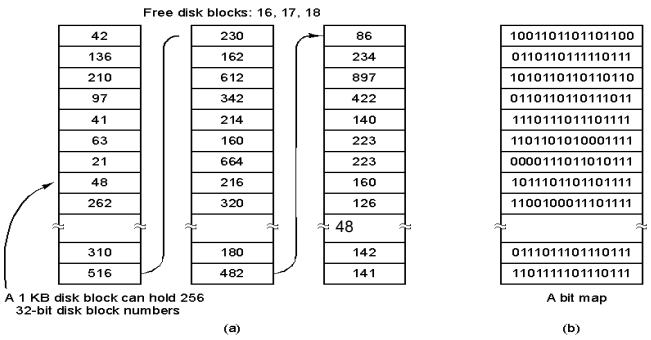
# Disk Space Management (1)




- Dark line (left hand scale) gives data rate of a disk
- Dotted line (right hand scale) gives disk space efficiency
- All files 2KB

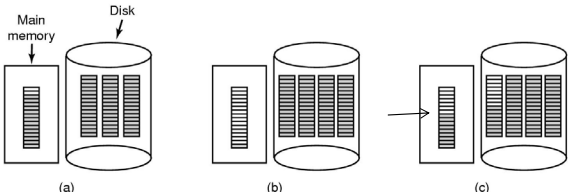


# Disk Space Management (2)




(a) Storing the free list on a linked list
(b) A bit map

# Disk Space Management (3)




(a) Almost-full block of pointers to free disk blocks in RAM

- three blocks of pointers on disk

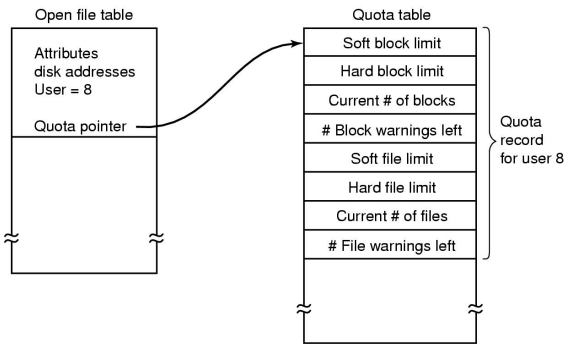
(b) Result of freeing a 3-block file

(c) Alternative strategy for handling 3 free blocks

- shaded entries are pointers to free disk blocks

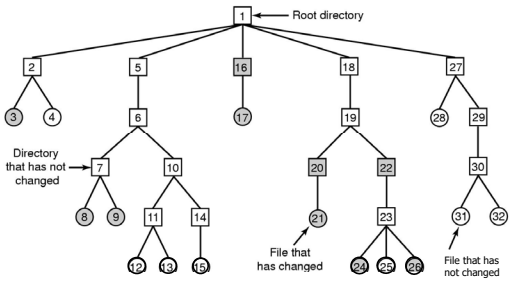


# Disk Space Management (4)




Quotas for keeping track of each user's disk use

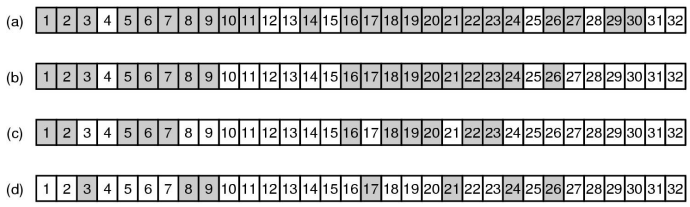
# File System Reliability (1)




- A file system to be dumped
  - squares are directories, circles are files
  - shaded items, modified since last dump
  - each directory & file labeled by i-node number

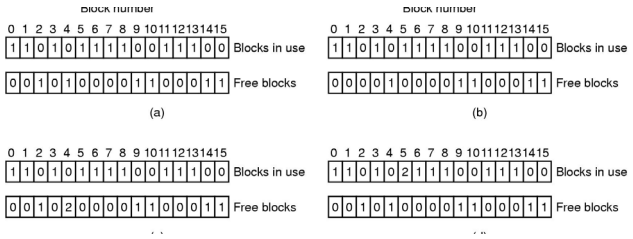


# File System Reliability (2)

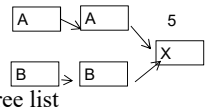



Bit maps used by the logical dumping algorithm

# File System Reliability (3)

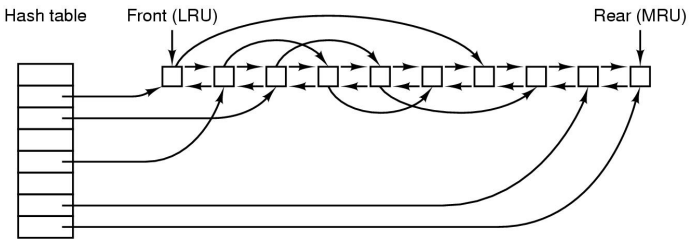



- File system states
  (a) consistent
  (b) missing block
  (c) duplicate block in free list
  (d) duplicate data block





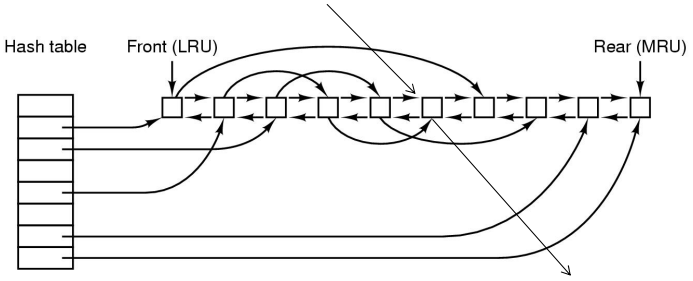

# File System Performance (1)




write- through vs. non-write-through

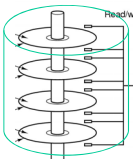
The block cache data structures

# File System Performance (1)




write- through vs. non-write-through

The block cache data structures



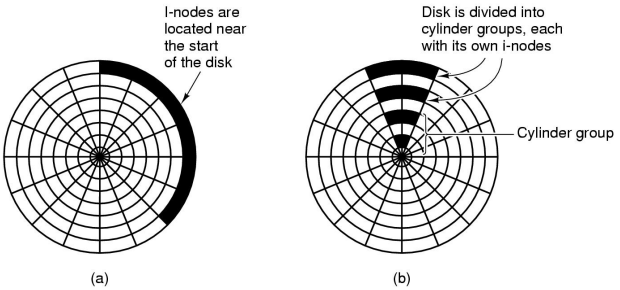


# buffer cache – זיכרון מטמון

H * T1 + (1-H) * (T1 + T2) = H*T1 + T1 – H*T1 + (1-H) * T2
=
= T1 + (1-H)*T2

H=hit rate ratio
1-H=miss rate ratio
T1=high level memory access time
T2=low level memory access time

# File System Performance (2)




- I-nodes placed at the start of the disk
- Disk divided into cylinder groups
  - each with its own blocks and i-nodes

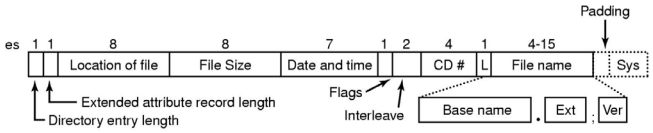
לעבור לשאלות
על הפרק



# Log-Structured File Systems

- With CPUs faster, memory larger
  - disk caches can also be larger
  - increasing number of read requests can come from cache
  - thus, most disk accesses will be writes
- LFS Strategy structures entire disk as a log
  - have all writes initially buffered in memory
  - periodically write these to the end of the disk log
  - when file opened, locate i-node, then find blocks

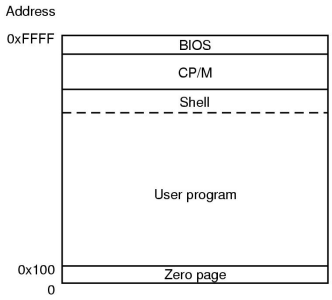
# Example File Systems
## CD-ROM File Systems




The ISO 9660 directory entry

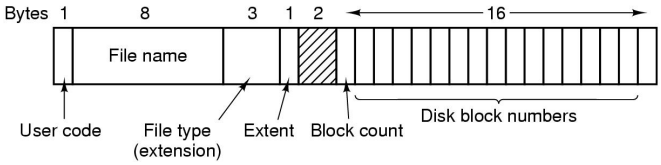


# The CP/M File System (1)




Memory layout of CP/M

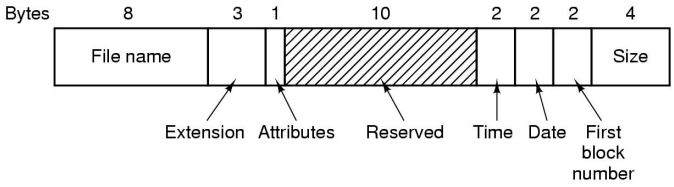
# The CP/M File System (2)




The CP/M directory entry format



# The MS-DOS File System (1)




The MS-DOS directory entry

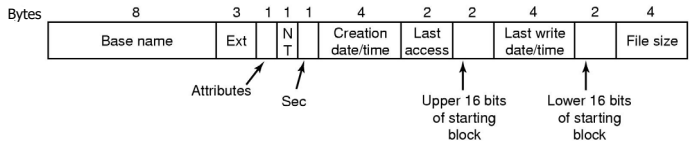
# The MS-DOS File System (2)

| Block size | FAT-12 | FAT-16 | FAT-32 |
|---|---|---|---|
| 0.5 KB | 2 MB | | |
| 1 KB | 4 MB | | |
| 2 KB | 8 MB | 128 MB | |
| 4 KB | 16 MB | 256 MB | 1 TB |
| 8 KB | | 512 MB | 2 TB |
| 16 KB | | 1024 MB | 2 TB |
| 32 KB | | 2048 MB | 2 TB |

- Maximum partition for different block sizes
- The empty boxes represent forbidden combinations

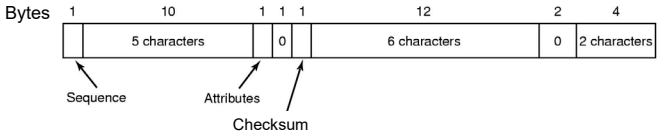


# The Windows 98 File System (1)




The extended MOS-DOS directory entry used in Windows 98

# The Windows 98 File System (2)




An entry for (part of) a long file name in Windows 98

15:15

47

# The Windows 98 File System (3)

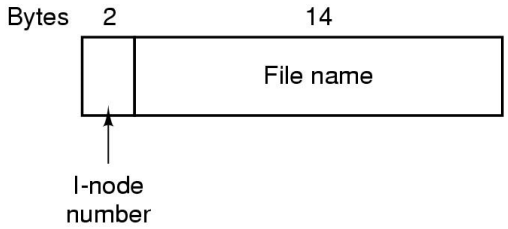
| Bytes | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 68 | d | o | g | | | | | A | 0 | CK | | | | | | | | | | | 0 | | | |
| 3 | o | v | e | | | | | A | 0 | CK | t | h | e | | | l | a | | | | 0 | z | y | |
| 2 | w | n | | | f | o | | A | 0 | CK | x | | | j | u | m | p | | | | 0 | s | | |
| 1 | T | h | e | | | | q | A | 0 | CK | u | i | c | k | | | | b | | | 0 | r | o | |
| T | H | E | Q | U | I | ~ | 1 | A | NT | S | Creation time | | | | Last acc | | Upp | Last write | | | Low | Size | | |

An example of how a long name is stored in Windows 98

15:15

48

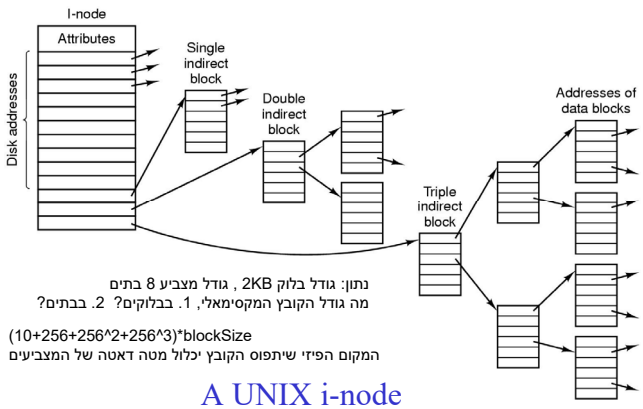
# The UNIX V7 File System (1)




A UNIX V7 directory entry



# The UNIX V7 File System (2)

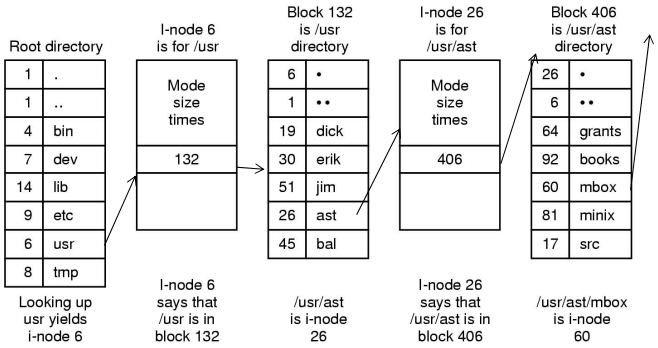



נתון: גודל בלוק 2KB , גודל מצביע 8 בתים
מה גודל הקובץ המקסימאלי, 1. בבלוקים? 2. בבתים?

(10+256+256^2+256^3)*blockSize
המקום הפיזי שיתפוס הקובץ יכלול מטה דאטה של המצביעים

A UNIX i-node

# The UNIX V7 File System (3)




The steps in looking up */usr/ast/mbox*



**שאלה 18**

מערכת הקבצים של מערכת הפעלה מסוימת משתמשת בשיטת ה i-node.

- גודל הבלוק במערכת הקבצים הוא 2 Kbyte
- כתובת הבלוק היא 8 בתים (bytes)

Block=256 pointer

- 10 שדות של ה i-node יכולים להחזיק ישירות כתובת הבלוק בדיסק
- שדה נוסף אחד נועד להחזיק כתובת של ה single indirect block
- עוד שדה נוסף אחד נועד להחזיק כתובת של ה double indirect block
- ועוד שדה נוסף אחד נועד להחזיק כתובת של ה triple indirect block

גודלו של קובץ מסוים במערכת 632 Kbyte. מהי כמות הבלוקים שדרושה להחזקת קובץ זה במערכת הקבצים (לא כולל את הבלוק שמכיל את ה i-node של הקובץ)?

632KB=316 blocks
10+(1)+256+(1+1)+50

א) 316
ב) 317
ג) 318
ד) 319
ה) 320

שאלה 1

מהו מספר פעולות הקריאה של בלוקים מהדיסק שצריך לבצע בהרצת השורה הזאת:

fd=open("/a/b/c");

במערכת הפעלה UNIX, כאשר:

- ניתן להניח כי קובץ c כבר קיים על הדיסק.
- גודלו של כל קובץ ספרייה הוא בלוק אחד.
- הבלוקים (למעט ה- super block והבלוק המכיל את ה- i-node של ספריית השורש) אינם נמצאים בזיכרון מראש.
- קריאות המערכת אינן נכשלות
- ה- i-node נמצאים בבלוקים שונים

15:15

53

תשובה

- במערכת ההפעלה UNIX ספריית השורש "/" נמצאת ב- i-node שמספרו 2. לפי ההנחה, הבלוק שמכיל i-node זה כבר נמצא בזיכרון. לכן הבאת הבלוק של ספריית השורש תצריך רק פעולת קריאה אחת.
- בבלוק של ספריית השורש בודקים מהו מספר ה- i-node של הספרייה a ומביאים את הבלוק המכיל אותה, ולאחר מכן – את הבלוק המכיל את קובץ הספרייה a עצמו. מספר הקריאות מהדיסק הוא 2.
- בבלוק של ספרייה a בודקים מהו מספר ה- i-node של הספרייה b ומביאים את הבלוק המכיל אותה, ולאחר מכן – את הבלוק המכיל את קובץ הספרייה b עצמו. מספר הקריאות מהדיסק הוא 2.
- בבלוק של ספרייה b בודקים מהו מספר ה- i-node של הקובץ c ומביאים את הבלוק המכיל אותה (כלומר, את הבלוק שמכיל את ה- i-node של c). מספר הקריאות מהדיסק הוא 1.

15:15

54

שאלה 2

במערכת קבצים נתונה משתמשים בשיטת i-nodes . נתון כי

- גודל הבלוק הוא 1 kbyte
- גודל המצביע לבלוק הוא 8 בתים (8 bytes)
- כל i-node מכיל עשרה מצביעים ישירים ושלושה מצביעים לא ישירים: single-indirect, double-indirect ו- triple-indirect (מצביע אחד מכל סוג)

ידוע כי קובץ מסוים משתמש ב- 21 בלוקים (כולל בלוקים של נתונים ובלוקים של מצביעים.

אין לוקחים בחשבון את הבלוק המכיל את ה-i-node עצמו).

מהו גודלו של הקובץ הזה?

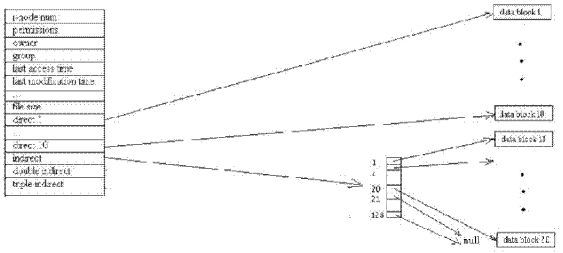


10 מצביעים ישירים מצביעים ישירות על 10 בלוקים של נתונים (מ- 1 עד 10). נותרו לנו עוד 11 בלוקים, שאחד מהם חייב להיות בלוק פנימי של מערכת הקבצים.

בלוק פנימי של מערכת הקבצים יכול להכיל

1kbyte/8Bytes = 2^7

מצביעים לבלוקים אחרים. במקרה שלנו, אלה בלוקים של נתונים מ- 11 עד 20). ראו האיור הבא:



שאלה 3

להלן תוצאות הבדיקה של עקביות מערכת הקבצים:

| Block number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Blocks in use | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| Free blocks | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

מה אפשר להסיק על פי הנתונים הנ"ל?



תשובה

- בלוק מס' 8 לא חסר. כדי להחזיר את מערכת הקבצים למצב עקבי, יש להוסיפו לרשימת הבלוקים הפנויים.
- בלוק מס' 10 הינו בלוק חופשי ומוקצה. כדי להחזיר את מערכת הקבצים למצב עקבי, יש להוציא את הבלוק מרשימת הבלוקים הפנויים.

שאלה 1

נתונות שלוש ספריות שהמידע שלהן נתון בטבלאות הבאות. מהו נתיב path תקני במערכת זו?

שים לב להבדל בין נתיב מוחלט מהשורש לנתיב יחסי.

תזכורת: ./ מציין נתיב יחסי מהספרייה הנוכחית.

directory

| | |
|---|---|
| 26 | • |
| 6 | •• |
| 64 | grants |
| 92 | books |
| 60 | mbox |
| 81 | minix |
| 17 | src |

directory

| | |
|---|---|
| 1 | . |
| 1 | .. |
| 4 | bin |
| 7 | dev |
| 14 | lib |
| 9 | etc |
| 6 | usr |
| 8 | tmp |

directory

| | |
|---|---|
| 6 | • |
| 1 | •• |
| 19 | dick |
| 30 | erik |
| 51 | jim |
| 26 | ast |
| 45 | bal |

א- /usr/ast/src

ב- ./usr/ast/src

ג- /grants/bin/dev

ד- ./grants/bin/dev

ה- /src/ast/usr

ו- ./src/ast/usr

15:15

59

**שאלה 10**

האם השימוש בזיכרון מטמון (Buffer Cash)מסוג write-trough מגדיל את חסינותה של מערכת הקבצים בהשוואה לשימוש בזיכרון מטמון מסוג non-write-trough?

א) כן. וזאת עקב ביצוע מהיר יותר של כתיבות בלוקים לדיסק.

ב) לא. וזאת עקב העיכוב בעדכון בלוקים בדיסק.

ג) לא. כוון שחסינותה של מערכת הקבצים איננה מושפעת כלל וכלל מזיכרון מטמון.

ד) אף תשובה קודמת איננה נכונה.

הערה:מערכת קבצים חסינה היא מערכת קבצים שלגביה קיים סיכוי קטן להישאר במצב של חוסר עקביות.

15:15 10/09/2023

60