

## File Systems

1

## 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

November 17, 2003

CS 3204: OS, Fall 2003

2

## File Naming

Extension	Meaning
file.bak	Backup file
file.c	C source program
file.gif	CompuServe Graphical Interchange Format image
file.hlp	Help file
file.html	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

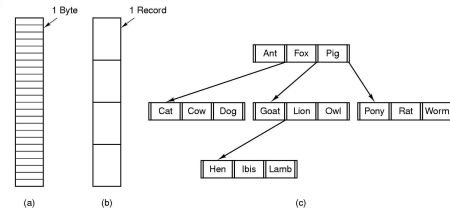
### Typical file extensions.

November 17, 2003

CS 3204: OS, Fall 2003

3

## File Structure



### Three kinds of files

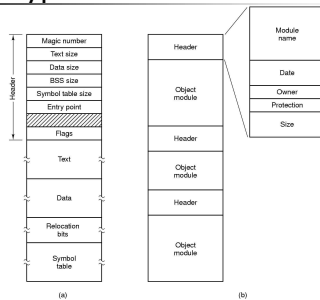
- byte sequence
- record sequence
- tree

November 17, 2003

CS 3204: OS, Fall 2003

4

## File Types



(a) An executable file (b) An archive

November 17, 2003

CS 3204: OS, Fall 2003

5

## 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

November 17, 2003

CS 3204: OS, Fall 2003

6

## 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

### Possible file attributes

November 17, 2003

CS 3204: OS, Fall 2003

7

## 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

November 17, 2003

CS 3204: OS, Fall 2003

8

## 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 */

```

November 17, 2003

CS 3204: OS, Fall 2003

9

## 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 /* error on last read */
    exit(5);
}

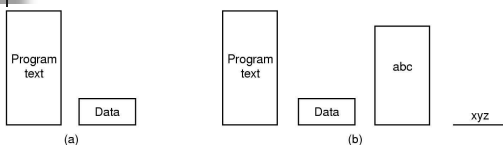
```

November 17, 2003

CS 3204: OS, Fall 2003

10

## Memory-Mapped Files



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

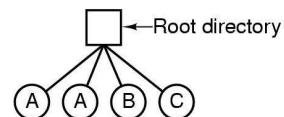
(b) Process after mapping existing file *abc* into one segment creating new segment for *xyz*

November 17, 2003

CS 3204: OS, Fall 2003

11

## Directories: Single-Level Directory Systems



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

November 17, 2003

CS 3204: OS, Fall 2003

12

## Two-level Directory Systems

Letters indicate *owners* of the directories and files

November 17, 2003 CS 3204: OS, Fall 2003 13

## Hierarchical Directory Systems

A hierarchical directory system

November 17, 2003 CS 3204: OS, Fall 2003 14

## Path Names

A UNIX directory tree

November 17, 2003 CS 3204: OS, Fall 2003 15

## Directory Operations

1. Create
2. Delete
3. Opendir
4. Closedir
5. Readdir
6. Rename
7. Link
8. Unlink

November 17, 2003 CS 3204: OS, Fall 2003 16

## File System Implementation

A possible file system layout

November 17, 2003 CS 3204: OS, Fall 2003 17

## Implementing Files (1)

(a) Contiguous allocation of disk space for 7 files  
 (b) State of the disk after files *D* and *E* have been removed

November 17, 2003 CS 3204: OS, Fall 2003 18

### Implementing Files (2)

Storing a file as a linked list of disk blocks

November 17, 2003 CS 3204: OS, Fall 2003 19

### Implementing Files (3)

Linked list allocation using a file allocation table in RAM

November 17, 2003 CS 3204: OS, Fall 2003 20

### Implementing Files (4)

An example i-node

November 17, 2003 CS 3204: OS, Fall 2003 21

### 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

November 17, 2003 CS 3204: OS, Fall 2003 22

### Implementing Directories (2)

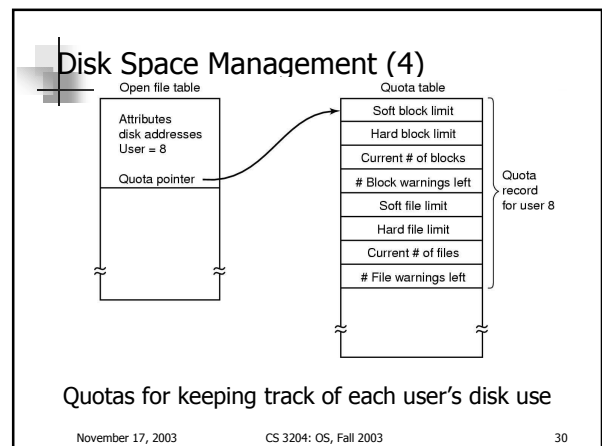
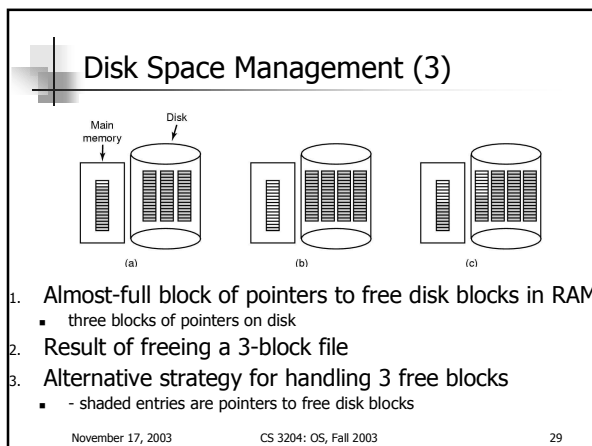
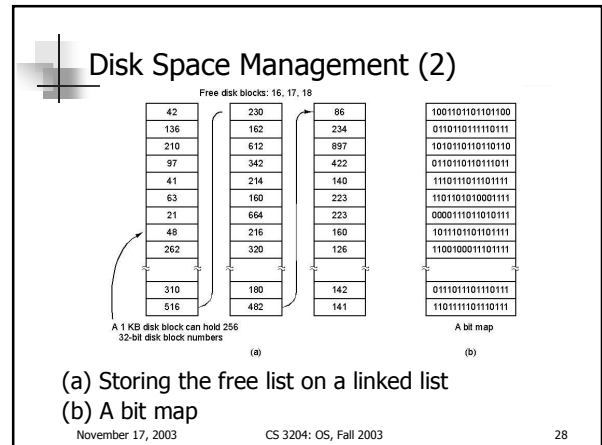
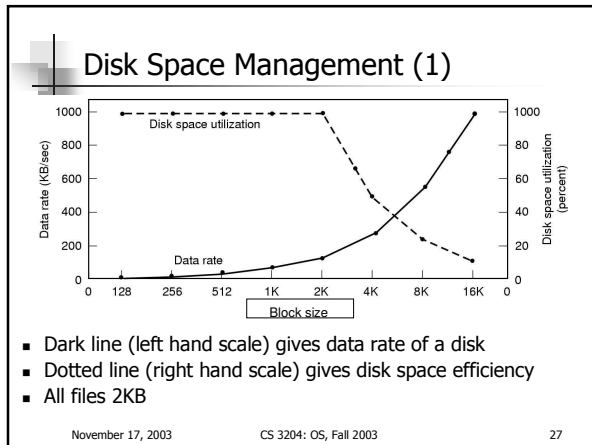
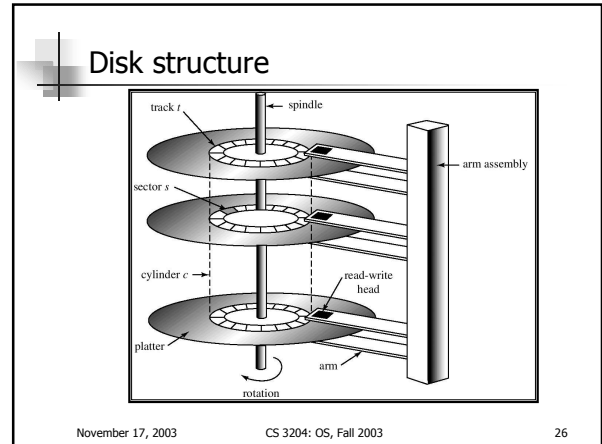
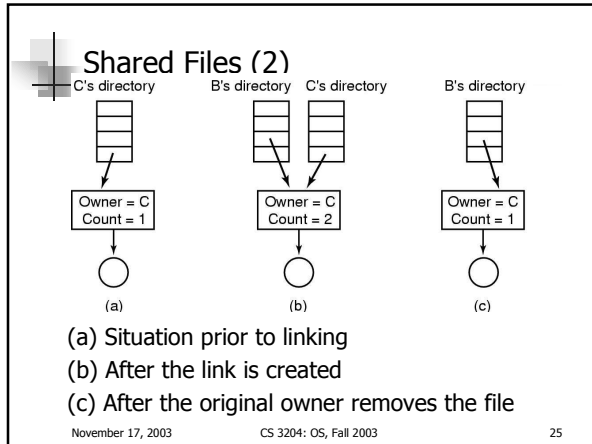
- Two ways of handling long file names in directory
  - (a) In-line
  - (b) In a heap

November 17, 2003 CS 3204: OS, Fall 2003 23

### Shared Files (1)

File system containing a shared file

November 17, 2003 CS 3204: OS, Fall 2003 24



## File System Performance (1)

The block cache data structures

November 17, 2003      CS 3204: OS, Fall 2003      31

## 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

November 17, 2003      CS 3204: OS, Fall 2003      32

## File System API Calls in Windows 2000 (1)

Win32 API function	UNIX	Description
CreateFile	open	Create a file or open an existing file; return a handle
DeleteFile	unlink	Destroy an existing file
CloseHandle	close	Close a file
ReadFile	read	Read data from a file
WriteFile	write	Write data to a file
SetFilePointer	lseek	Set the file pointer to a specific place in the file
GetFileAttributes	stat	Return the file properties
LockFile	fcntl	Lock a region of the file to provide mutual exclusion
UnlockFile	fcntl	Unlock a previously locked region of the file

- Principle Win32 API functions for file I/O
- Second column gives nearest UNIX equivalent

November 17, 2003      CS 3204: OS, Fall 2003      33

## File System API Calls in Windows 2000 (2)

```

/* Open files for input and output. */
inhandle = CreateFile("data", GENERIC_READ, 0, NULL, OPEN_EXISTING, 0, NULL);
outhandle = CreateFile("newf", GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL);

/* Copy the file. */
do {
    s = ReadFile(inhandle, buffer, BUF_SIZE, &count, NULL);
    if (s && count > 0) WriteFile(outhandle, buffer, count, &ocnt, NULL);
} while (s > 0 && count > 0);

/* Close the files. */
CloseHandle(inhandle);
CloseHandle(outhandle);
  
```

**A program fragment for copying a file using the Windows 2000 API functions**

November 17, 2003      CS 3204: OS, Fall 2003      34

## File System API Calls in Windows 2000 (3)

Win32 API function	UNIX	Description
CreateDirectory	mkdir	Create a new directory
RemoveDirectory	rmdir	Remove an empty directory
FindFirstFile	opendir	Initialize to start reading the entries in a directory
FindNextFile	readdir	Read the next directory entry
MoveFile	rename	Move a file from one directory to another
SetCurrentDirectory	chdir	Change the current working directory

- Principle Win32 API functions for directory management
- Second column gives nearest UNIX equivalent, when one exists

November 17, 2003      CS 3204: OS, Fall 2003      35

## File System Structure (1)

**The NTFS master file table**

November 17, 2003      CS 3204: OS, Fall 2003      36

## File System Structure (2)

Attribute	Description
Standard information	Flag bits, timestamps, etc.
File name	File name in Unicode; may be repeated for MS-DOS name
Security descriptor	Obsolete. Security information is now in \$Extend\$Secure
Attribute list	Location of additional MFT records, if needed
Object ID	64-bit file identifier unique to this volume
Reparse point	Used for mounting and symbolic links
Volume name	Name of this volume (used only in \$Volume)
Volume information	Volume version (used only in \$Volume)
Index root	Used for directories
Index allocation	Used for very large directories
Bitmap	Used for very large directories
Logged utility stream	Controls logging to \$LogFile
Data	Stream data; may be repeated

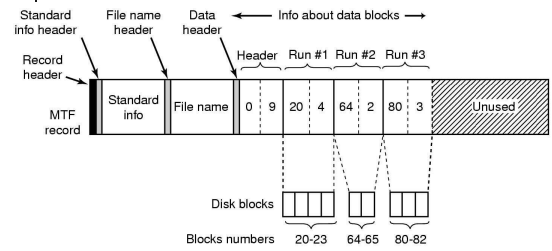
The attributes used in MFT records

November 17, 2003

CS 3204: OS, Fall 2003

37

## File System Structure (3)



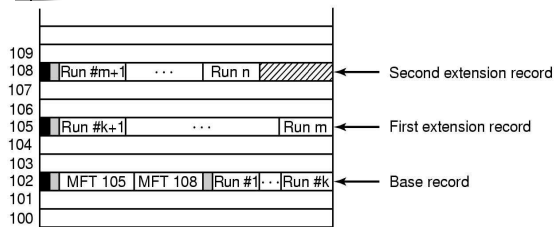
An MFT record for a three-run, nine-block file

November 17, 2003

CS 3204: OS, Fall 2003

38

## File System Structure (4)



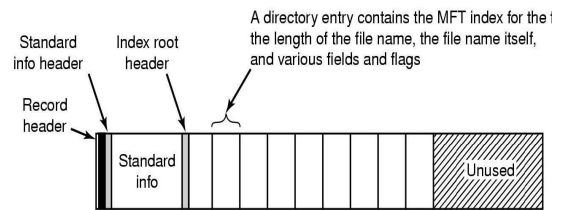
A file that requires three MFT records to store its runs

November 17, 2003

CS 3204: OS, Fall 2003

39

## File System Structure (5)

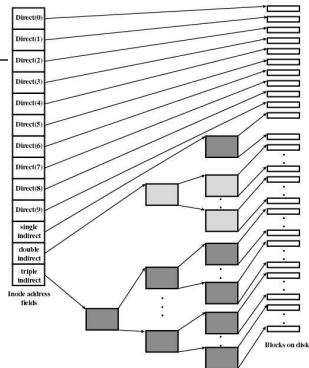


The MFT record for a small directory.

November 17, 2003

CS 3204: OS, Fall 2003

40



November 17, 2003

Figure 12.13 UNIX Block Addressing Scheme  
CS 3204: OS, Fall 2003

41