

File system

OS 2005-06 1

Goal of the file system

- Uniform naming
- Directories (containing files)
- File extension (e.g. .c, .h, .cpp) logically enforcing file using
 - Windows is aware of extension creating associations between applications and extensions
- File structure
 - Blocks:
 - 1 byte (sequence of bytes, not blocks for real)
 - Records (blocks for real)
 - Sometimes, tree-like organization of records

OS 2005-06 2

Types

- Regular files
 - Our data
- Directories
 - System files for maintaining group of files
- Character/block special files
- Pipes
 - Special files
- ASCII/binary differences

OS 2005-06 3

Example (Unix executable)

header {

Magic number
Text size
Data size
BSS size
Symbol table size
Entry point
Flags
Text
Data
Relocation bits
Symbol table

OS 2005-06 4

About files

- We've already seen this:
 - File access: random, sequential
 - Seek operation
 - Attributes:
 - Read/write permission
 - Owner
 - Time of creation, last access, etc.
 - Archive (for backups)
 - Operations:
 - Read, write, open, close, creation, deletion, get/set attributes, rename (w/out copy)

OS 2005-06 5

Memory mapped files

- Map a file into a part of the process address space that opens it
 - Convenient:
 - I/O becomes memory access
 - Paging becomes the read/write mechanism
 - Troubles:
 - How big is the file?
 - How to deal with a file w/ holes (should the OS map all addresses?)
 - Need a mechanism to ask for frequent "real" write to disk, otherwise the file is not written until a page is evicted
 - Imagine that your word processor crashes and the page hasn't been saved in the last couple of hours!

OS 2005-06 6

Directories

- Single level directory
 - Usually on embedded systems
- Two-level directory system
 - Old
- Hierarchical directory system
 - The usual thing everyone is familiar with
 - Multiple/single root (Windows/Unix)

OS 2005-06

7

Path names

- Already seen:
 - Delimiters / or \ (win)
 - Current directory (relative path names)
- Directory operations
 - Create, delete, opendir/closedir, readdir, rename, link/unlink
 - “Link” as seen earlier
- Mount (Unix)
 - It exists a similar concept in Win2K server

OS 2005-06

8

Implementation

OS 2005-06

9

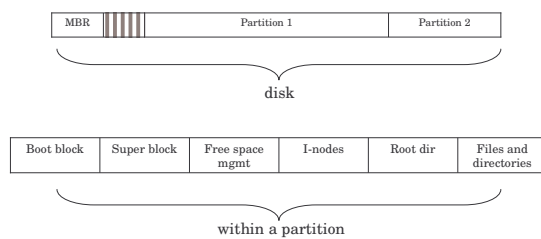
Layout

- Stored on disk, how the file system is organized
- Partitions
 - Disks are divided into partitions w/ independent file system in each
- MBR → Sector 0, where the computer boots from
 - The end of the MBR contains the partition table
- One of the partition is marked as active
- When the computer is booted, the BIOS loads and execute the MBR. The program (MBR) locates the active partition, reads the first block (called *boot block*) and executes it. The program in the boot block is the OS loader, knows where the kernel image is and how to run it appropriately.

OS 2005-06

10

Idealized



OS 2005-06

11

Implementing files

- Approximately as tracking and allocating memory
 - Same spatial organization
 - Disk divided in blocks (similar to the concept of pages)
 - Blocks do not need to be the same size as physical sectors (they're the abstraction of the OS)
- As for memory
 - Internal fragmentation (as for memory)
 - External fragmentation (if we try to allocated blocks contiguously)

OS 2005-06

12

Contiguous allocation

- Fragmentation
 - Files and holes
- Read time excellent:
 - Single seek operation (beginning of file)
 - Then read contiguously
- Disk compaction is very slow
 - It can be done but it takes ages (in computer terms)

OS 2005-06

13

Imagine the consequences...

- You start preparing the file for your thesis and the word processor asks for the final size in bytes!
 - You choose 100Mbytes, maybe there's no such hole in the disk. No thesis
 - You ask for 1Mbyte. You write up to page 10 and "save as..." fails (the hole was too small)
- Contiguous allocation is used though
 - CD-ROMs, because we know the size of files in advance

OS 2005-06

14

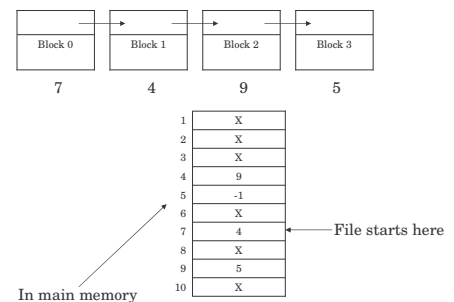
Linked list allocation

- Linked list:
 - Each block contains also the pointer to the next block (or -1 if last block of file)
 - Sequential read is fine if starting from block 1 of the file
 - Random access painful
 - Also, the room for the pointer changes the size of blocks. The amount of storage is no longer a power of two (can slow down things)

OS 2005-06

15

Linked list w/ table in mem



OS 2005-06

16

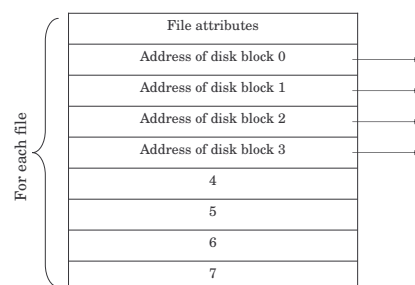
FAT in memory?

- 20GB disk, 1Kbyte block size
- 20M entries
- 3bytes each, 4 for efficiency (32bit processor)
- 80Mbytes of RAM
- If paged, still 80Mbytes of virtual memory + the bus traffic due to paging

OS 2005-06

17

I-node solution



OS 2005-06

18

I-nodes

- Size of table
 - FAT: proportional to N, if disk has N blocks
 - I-nodes: proportional to the number of files open simultaneously
- Additionally
 - If the I-node is filled the last pointer is reserved for holding the address of the next block of I-nodes (another table similar to the first one)

OS 2005-06

19

Directories

- The directory entry contains the information about the files
 - E.g. where I-nodes are stored
- Where are attributes stored (creation times, permission, etc.)
 - In the directory itself (MSDOS)
 - In the I-node (Unix)
- Issues with:
 - Storing long file names
 - Searching large directories (over 1000 of files)
 - Linear search
 - Hash table based search

OS 2005-06

20

Shared files

- To show the same file as appearing in multiple directories
 - Note! The same file, not a copy
- If the directory structure contains only the pointer to the I-node (together with the file name)
 - Sharing means setting the pointer to the correct I-node
- Second solution. Having a special file of type LINK (symbolic linking)
 - In practice a redirection of the access to the shared file

OS 2005-06

21

Issues with shared files

- Accounting appropriately
 - What if the owner of the file deletes it but a link is still active on the file
 - Owner doesn't own a file but he/she's still charged for it
- LINK approach is a bit slower
 - The path must be followed and the correct I-node found

OS 2005-06

22

Disk space management

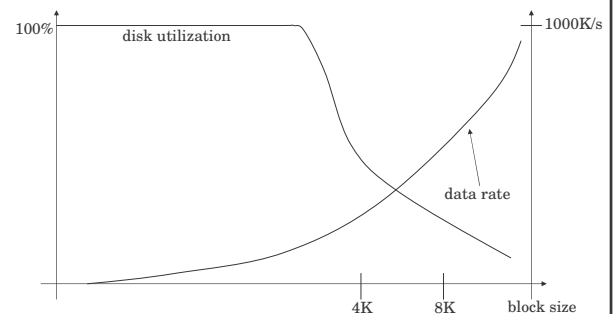
- As for memory
 - External vs. internal fragmentation
- Example:
 - Average seek time 10ms
 - Rotation time 8.33ms
 - Bytes per track 131072
 - Reading k bytes:

$$10 + (8.33/2) + (8.33/131072)*k$$

OS 2005-06

23

Choosing block size



OS 2005-06

24

Keeping track of free blocks

- As for memory
 - Linked list of free blocks
 - 16Gb disk → 16K pointers for a 1Kb block and 32bit block numbers
 - $16G = 2^{34}$, 1K block size means 2^{24} blocks. Each block contains 255 ~ 2^8 pointers (32bit each). 2^{16} blocks are required
 - Bitmap
 - Same disk, 2^{24} bits which requires:
 - $2^{(24 - 4 - 10)} = 2^{10}$
 - The bitmap is much smaller than the linked list
 - Usually the bitmap can be in main memory (at least a page)
 - Also for the linked list, for speeding up access part of it should be in main memory

OS 2005-06

25

Quotas

- Limit the disk space used by a single user
- Keep track of what files belong to each user
- ...just to know that it exists!

OS 2005-06

26

Backups

- Recover from
 - Stupidity
 - Disaster
- Physical dump
 - Sector by sector copy of the disk
- Logical dump
 - Backup software parser the directory tree and copies files (excluding /dev, pipes, etc.)

OS 2005-06

27

Backups

- Full or incremental
 - Full: copy everything (directories and files)
 - Incremental: copy only modified files + part of the directory hierarchy containing them
- Costly operations
- People are not aware of the full value of their data until they lose them

OS 2005-06

28

File system consistency

- Check whether the list of free blocks and files are mutually consistent
- Reliability
 - RAID (we've seen it)
 - Stable storage (algorithmically safe)
- Safety
 - What, where, and how often to backup files

OS 2005-06

29

Performance

- Caching
 - How to choose what to evict from cache
 - Similar to memory: FIFO, second chance, LRU can be used
 - ...but unfortunately, we shouldn't be caching forever. Avoid data not written for long time, just in case of a crash
 - Unix has a daemon saving to disk every 30 seconds or so (update)
 - MSDOS uses a write-through cache, "write" are scheduled as soon as possible (always consistent)
 - Windows started to use the first strategy too (more efficient)

OS 2005-06

30

Performance

- Block read ahead
 - Try to guess what's needed next
 - Try to estimate how sequential a file is accessed
 - If sequential, try reading ahead before blocks are needed
- Reducing arm motion
 - Where to put I-nodes
 - Try to do block clustering

The end