CS323 Operating Systems
File System IV

Yuanyuan Zhou

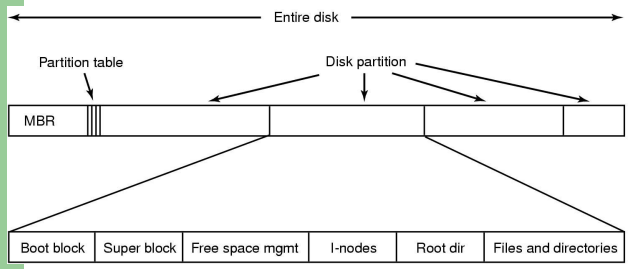
Lecture 26

4/2/2003

Content of this lecture

- Administrative announcements
- File systems implementation
- Summary

File System Implementation

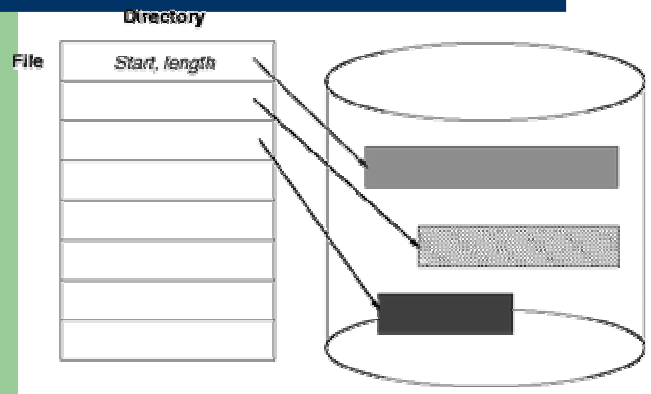


A possible file system layout

Allocation of Disk Space

- Low level access methods depend upon the disk allocation scheme used to store file data
  - Contiguous allocation
  - Linked list allocation
  - Block allocation

Contiguous Allocation



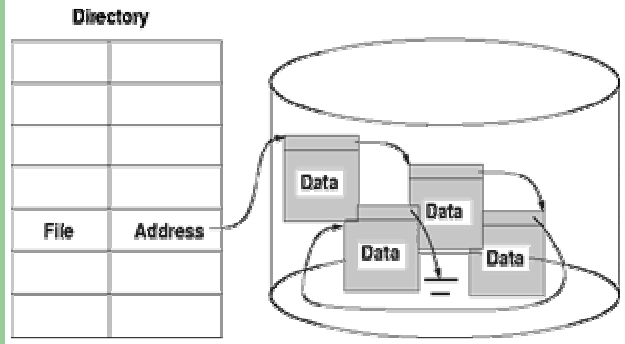
Contiguous Allocation Issues

- Access method suits sequential and direct access
- Directory table maps files into starting physical address and length
- Easy to recover in event of system crash
- Fast, often requires no head movement and when it does, head only moves one track

Contiguous Allocation Issues

- File is allocated large contiguous chunks
- User knows size of the file
- Expanding the file requires copying
- Dynamic storage allocation - first fit, best fit
- External fragmentation occurs on disk
- Users tend to overestimate space => internal fragmentation

Linked Allocation



Linked List Allocation Issues

- Each file is a linked list of chunks
- Pointers in list are not accessible to user
- Directory table maps files into head of list for a file
- A node in the list can be a fixed size physical block or a contiguous collection of blocks
- Easy to use - no estimation of size necessary

Linked List Allocation Issues

- Can grow in middle and at ends
- Space efficient, little fragmentation
- Slow - defies the principle of locality. Need to read through linked list nodes sequentially to find record of interest blocks
- Suited for sequential access but not direct access

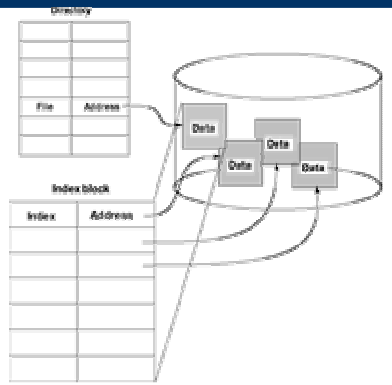
Linked List Allocation Issues

- Disk space must be used to store pointers (if disk block is 512 bytes, and disk address requires 4 bytes, then the user sees blocks of 508 bytes)
- Not very reliable. System crashes can scramble files being updated
- Important variation on linked allocation method: `file-allocation table' (FAT) - OS/2 and MS-DOS

Linked List Allocation Issues

- Summary: linked allocation solves the external fragmentation and size-declaration problems of contiguous allocation,
- However, it can't support efficient direct access

Indexed Allocation



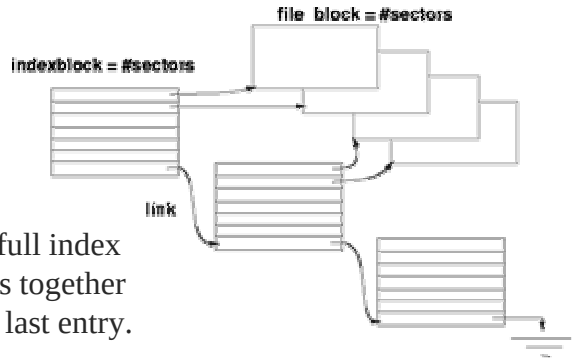
Indexed Allocation

- Solves external fragmentation
- Supports sequential, direct and indexed access
- Access requires at most one access to index block first. This can be cached in main memory

Indexed Allocation

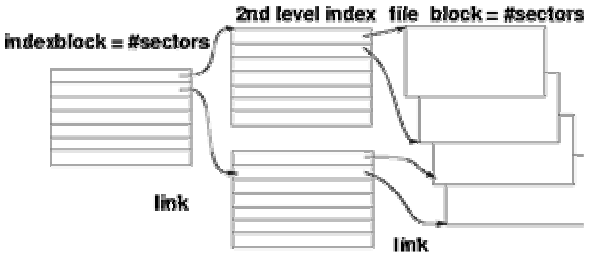
- File can be extended by rewriting a few blocks and index block
- Requires extra space for index block, possible wasted space
- Extension to big files issues

Other Forms of Indexed File Linked



Link full index blocks together using last entry.

Other Forms of Indexed File - Multilevel Index

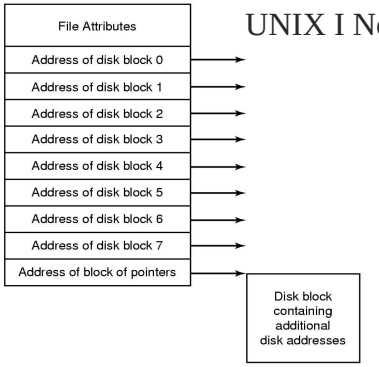


Multiple levels of index blocks

Discussion

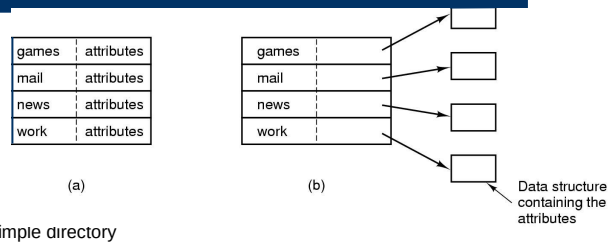
- Tradeoff between
  - Contiguous
  - Linked allocation
  - Indexed allocation

Implementing Files



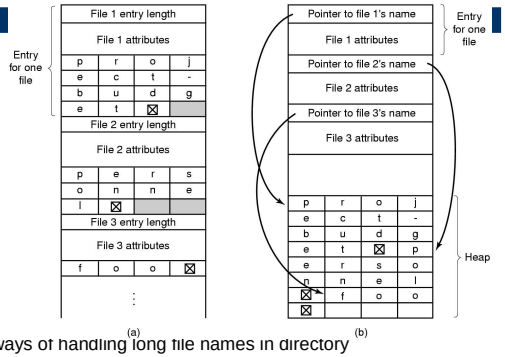
UNIX I Node

Implementing Directories (1)



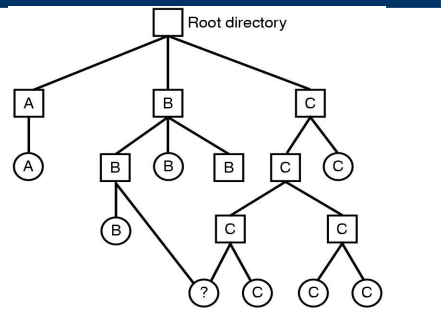
(a) A simple directory
fixed size entries
disk addresses and attributes in directory entry
(b) Directory in which each entry just refers to an i-node

Implementing Directories (2)



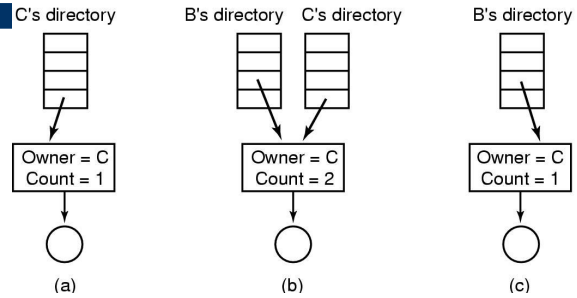
- Two ways of handling long file names in directory (a) In-line or (b) In a heap

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

Free Space Management

- Bit vector
  - A bit map is kept of free blocks
  - Each bit in a vector represents one block
  - If the block is free, the bit is zero
  - Simple to find n consecutive free blocks
  - Overhead is bit map
  - Example BSD file system

Free Space Management

- Free list
  - Keep a linked list of free blocks
  - Not very efficient because linked list needs traversal
  - Example system V R1

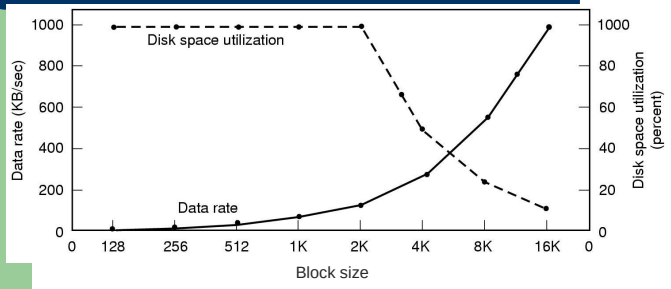
Free Space Management

- Linked list of indices
  - A linked list of index blocks is kept
  - Each index block contains addresses of free blocks and a
  - Pointer to the next index block
- A large number of free blocks can be found quickly

Free Space Management

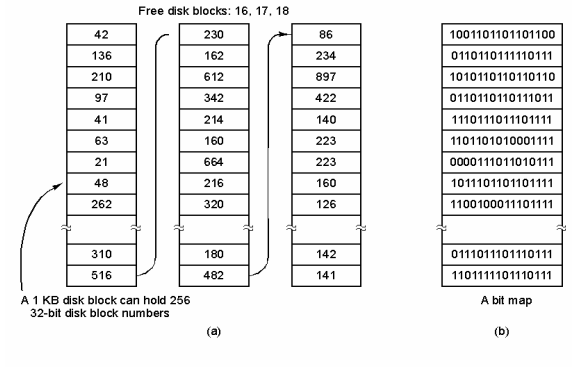
- Linked list of contiguous blocks that are free
  - The free list node consists of a pointer and the number of free blocks starting from that address
  - Blocks are joined together into larger blocks as necessary

Disk Space Management (1)

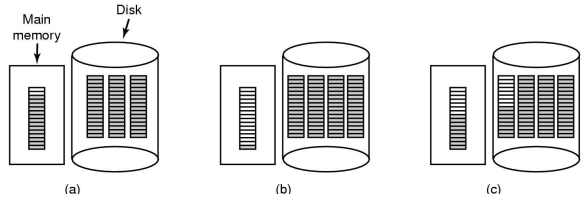


- Dark line (left hand scale) gives data rate of a disk
- Dotted line (right hand scale) gives disk space efficiency
- All files 2KB

Disk Space Management (2)



Disk Space Management (3)

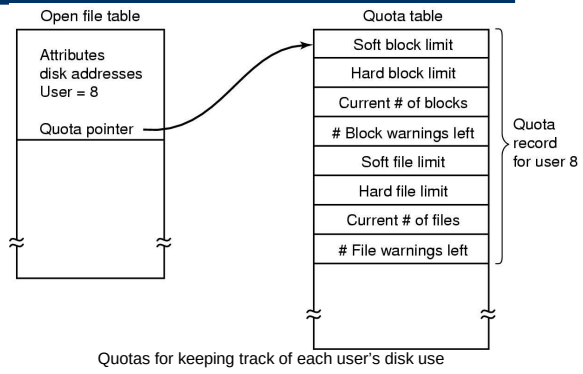


(a) Almost-full block of pointers to free disk blocks in RAM
- three blocks of pointers on disk

(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