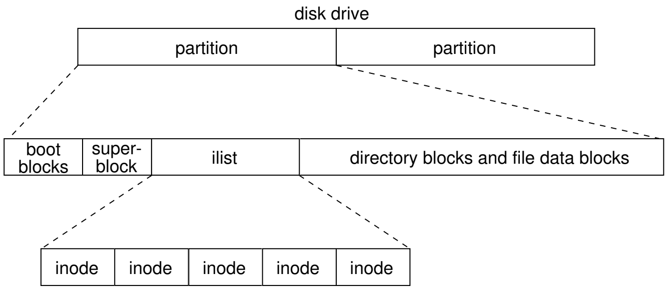
UNIX File System (Review)

- High-level view:



- Low-level view:

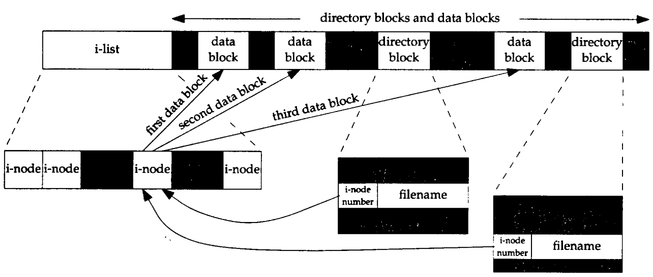


Diagram from *Advanced Programming in the UNIX Environment*, W. Richard Stevens, Addison Wesley, 1992.

Working with Directories

- Searching a directory in UNIX:
  - If filename begins with " / ", start at root of the file system tree (inode 2)
  - If filename begins with " ~ ", start at the user's home directory
  - If filename begins with any other character, start at current working directory
- Working directories
  - A file name can be given as the full *pathname*, separating levels by " / "
  - UNIX also keeps track of the inode number of current working directory for each process; we don't have to use full names
- A UNIX directory has two special entries
  - " . " refers to the directory itself
  - " .. " refers to the parent directory

Working with Directories (Lookup)

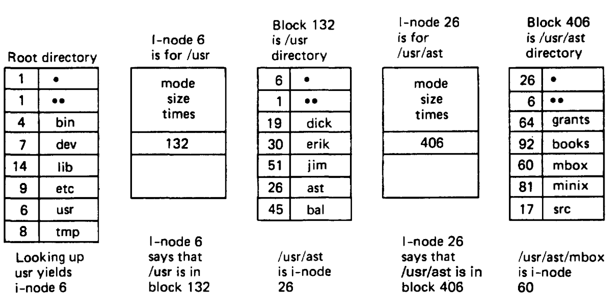


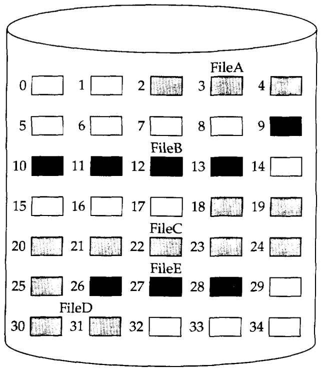
Fig. 4-16. The steps in looking up */usr/ast/mbox*.

- A directory is a table of entries:
  - 2 bytes — inumber
  - 14 bytes — file name (improved in BSD 4.2 and later)
- Search to find the file begins with either root, or the current working directory
  - Inode 2 points to the root directory (" / ")
  - Example above shows lookup of /usr/ast/mbox

Working with Directories (Links)

- UNIX supports links — two directories containing the same file
  - Example: aos/nachos & os/nachos
- Hard links (" ln *target_file directory* ")
  - Specified directory refers to the target file
    - Both directories point to same inode
  - Link count in inode is used to ensure that the file is deleted only when the last directory entry referring to it is removed
- Soft / symbolic links (" ln –s *target_file directory* ")
  - Adds a pointer to the target file (or target directory) from the specified directory
    - Special bit is set in inode, and the file just contains the name of the file it's linked to
    - View symbolic links with "ls –F" and "ls –l"
  - Can link across disk drives

Organization of Files (Contiguous Allocation)



File Allocation Table

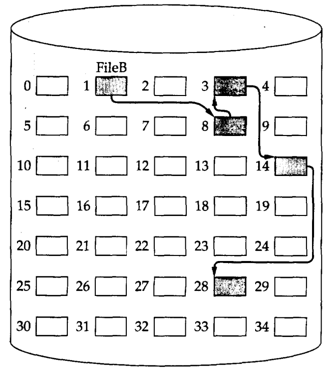
File Name	Start Block	Length
FileA	2	3
FileB	9	5
FileC	18	8
FileD	30	2
FileE	26	3

FIGURE 11.7 Contiguous file allocation

Diagram from *Operating Systems*, William Stallings, Prentice Hall, 1995.

- OS keeps an ordered list of free blocks
  - Allocates contiguous groups of blocks when it creates a file
  - File descriptor must store start block and length of file
- Used in IBM 370, some write-only disks

Organization of Files (Linked / Chained Allocation)



File Allocation Table

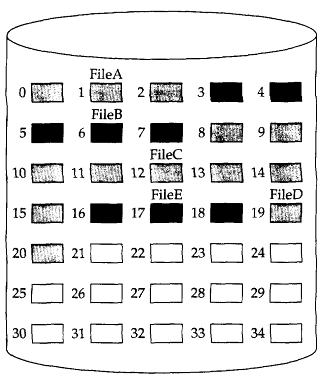
File Name	Start Block	Length
...	...	...
FileB	1	5
...	...	...

FIGURE 11.9 Chained allocation

Diagram from *Operating Systems*, William Stallings, Prentice Hall, 1995.

- OS keeps an ordered list of free blocks
  - File descriptor stores pointer to first block
  - Each block stores pointer to next block
- Used in DEC TOPS-10, Xerox Alto

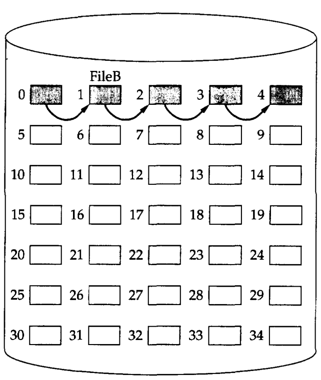
Organization of Files (Compaction for Contiguous and Linked Allocation)



File Allocation Table

File Name	Start Block	Length
FileA	0	3
FileB	3	5
FileC	8	8
FileD	19	2
FileE	16	3

FIGURE 11.8 Contiguous file allocation (after compaction)



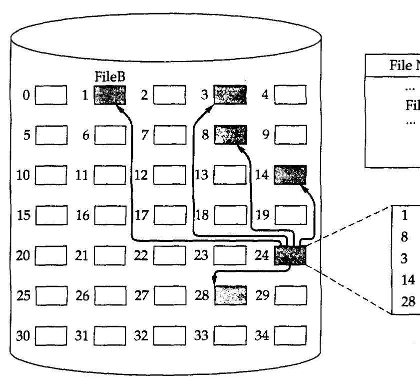
File Allocation Table

File Name	Start Block	Length
...	...	...
FileB	0	5
...	...	...

FIGURE 11.10 Chained allocation (after consolidation)

Diagrams from *Operating Systems*, William Stallings, Prentice Hall, 1995.

Organization of Files (Indexed Allocation)



File Allocation Table

File Name	Index Block
...	...
FileB	24
...	...

FIGURE 11.11 Indexed allocation with block portions

Diagram from *Operating Systems*, William Stallings, Prentice Hall, 1995.

- OS keeps a list of free blocks
  - OS allocates an array (called the index block) to hold pointers to all the blocks used by the file
  - Allocates blocks only on demand
  - File descriptor points to this array
- Used in DEC VMS, Nachos

Organization of Files (Multilevel Indexed Allocation)

- Used in UNIX (numbers below are for traditional UNIX, BSD UNIX 4.1)
- Each inode (file descriptor) contains 13 *block pointers*
  - First 10 pointers point to data blocks (each 512 bytes long) of a file
    - If the file is bigger than 10 blocks (5,120 bytes), the 11th pointer points to a *single indirect block*, which contains 128 pointers to 128 more data blocks (can support files up to 70,656 bytes)
      - If the file is bigger than that, the 12th pointer points to a *double indirect block*, which contains 128 pointers to 128 more single indirect blocks (can support files up to 8,459,264 bytes)
        - » If the file is bigger than that, the 13th pointer points to a *triple indirect block*, which contains 128 pointers to 128 more double indirect blocks
  - Max file size is 1,082,201,087 bytes

Organization of Files (Multilevel Indexed Allocation) (cont.)

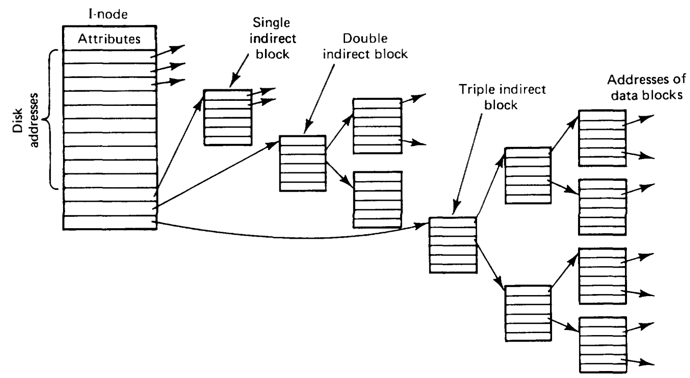


Diagram from *Modern Operating Systems*, Andrew Tanenbaum, Prentice Hall, 1992.

- BSD UNIX 4.2, 4.3:
  - Maximum block size is 4096 bytes
  - Inode contains 14 block pointers
    - 12 to data
    - 13 to single indirect block containing 1024 pointers, 14 to triple indirect block…
  - Max file size is 2^{32} bytes