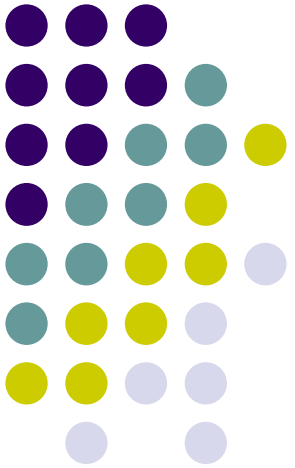
Operating System

Nguyen Tri Thanh

ntthanh@vnu.edu.vn



File System Implementation

File-System Structure

File-System Implementation

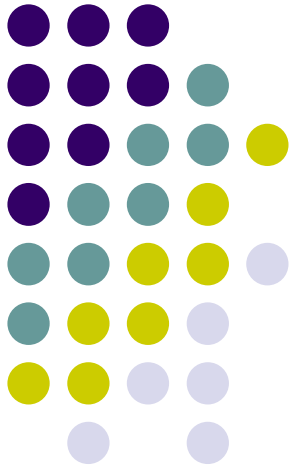
Directory Implementation

Allocation Methods

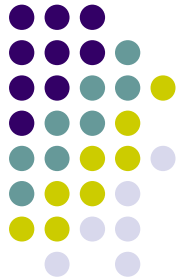
Free-Space Management

Efficiency and Performance

Recovery

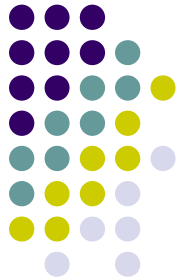


Objectives



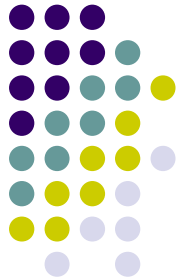
- Introduce what a file is
- Introduce file system implementation
- Introduce directory implementation
- Introduce 3 allocation methods
- Introduce free space management

Reference



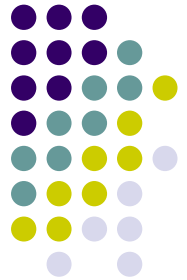
- Chapter 10, 11 of Operating System Concepts

File Concept



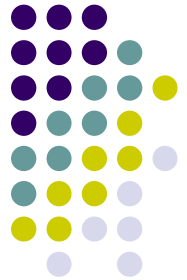
- Contiguous logical address space
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program

File Structure



- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters
- Who decides:
  - Operating system
  - Program

File Attributes



- Name
  - only information kept in human-readable form
- Identifier
  - unique (number) identifies file within file system
- Type
  - needed for systems that support different types
- Location
  - pointer to file location on storage device

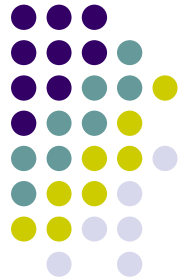
File Attributes (cont'd)

- Size
  - current file size
- Protection
  - controls who can do reading, writing, executing
- Time, date, and user identification
  - data for protection, security, and usage monitoring
- Information about files are kept in the directory structure on the disk

File System Structure

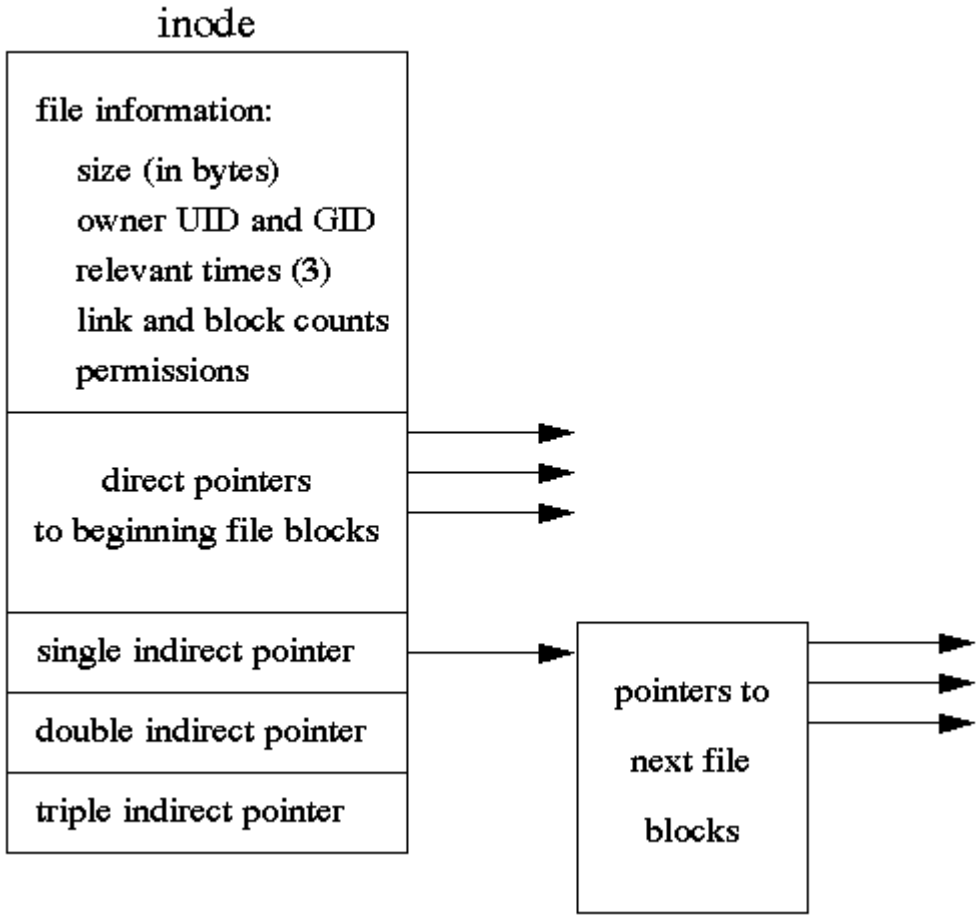
- File system resides on secondary storage
  - disks, CD ROM, DVD, flash drive,…
- File system organized into layers
- File control block (FCB)
  - storage structure of information about a file
  - *inode* (index node) is an implementation of FCB on Linux

A Typical File Control Block

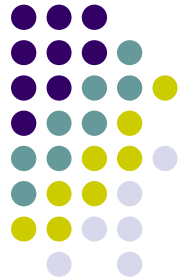


file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

Inode on Linux

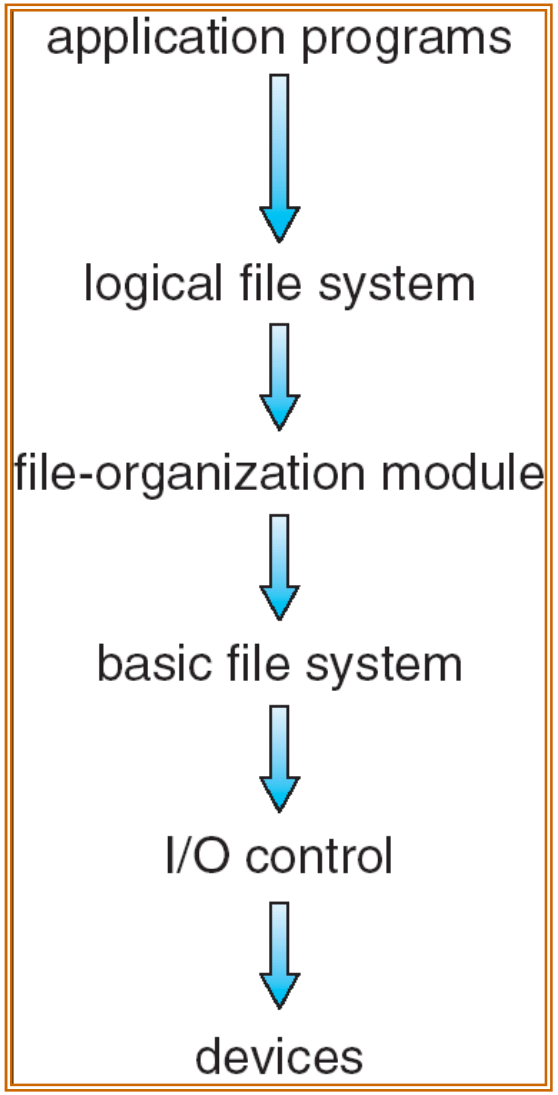


Question

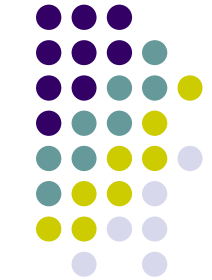


- Which of the following is incorrect about file control block (FCB)?
  A. a data structure containing information of a file
  B. OS needs a FCB to access a file
  C. FCB of a file is updated when a file is accessed
  D. FCBs reside in memory

Layered File System

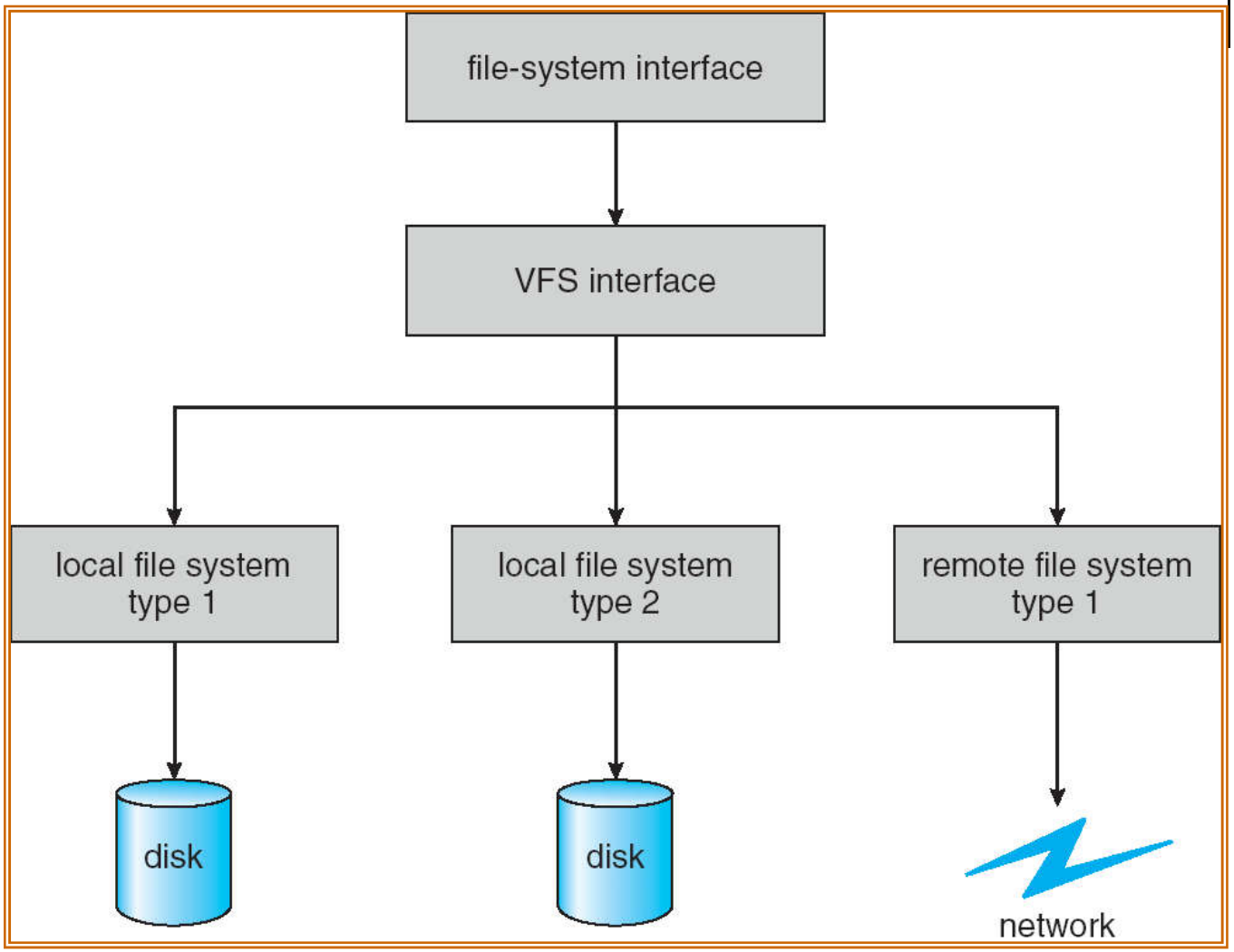


Virtual File Systems

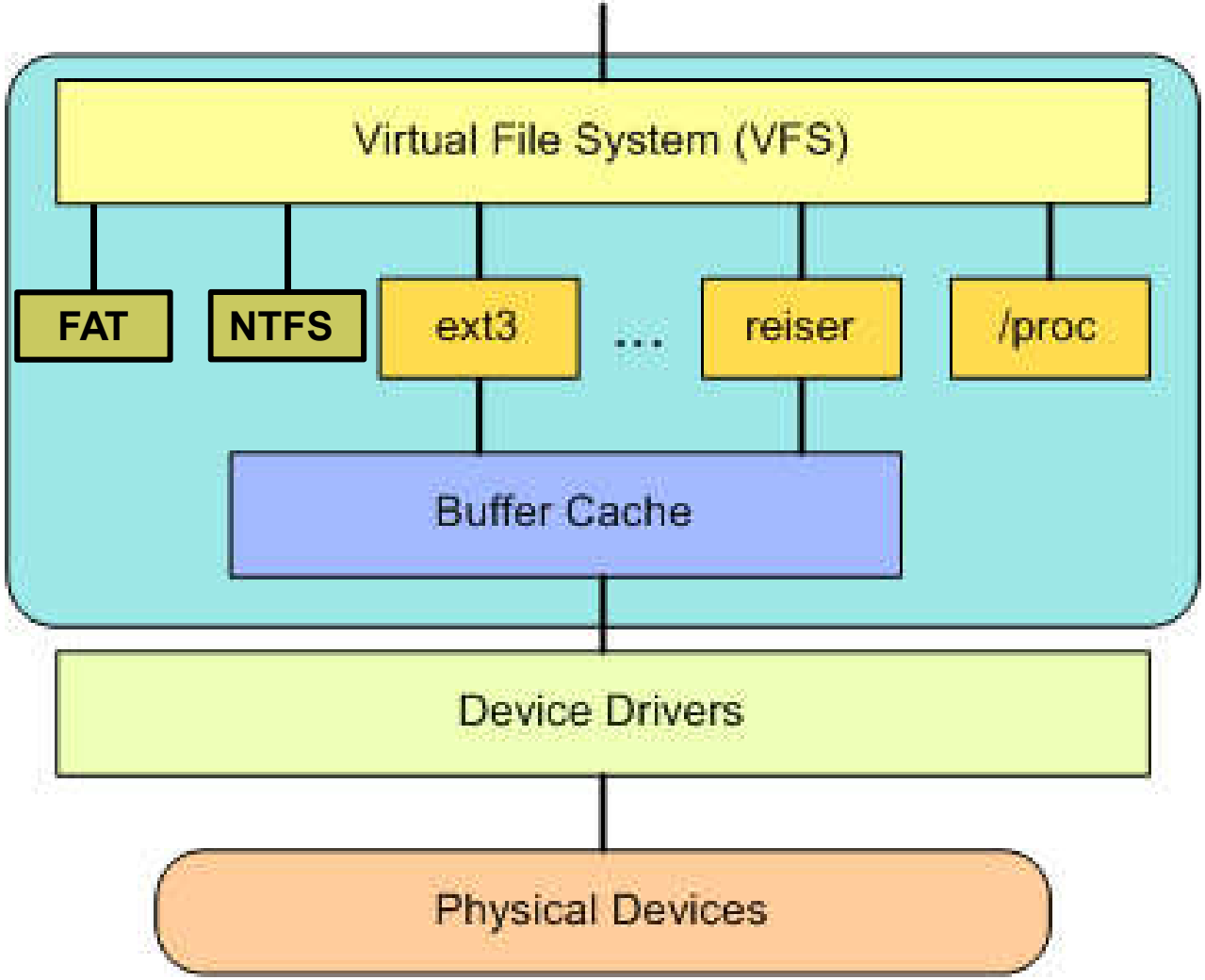


- Virtual File Systems (VFS)
  - provide an object-oriented way of implementing file systems
  - allow the same system call interface (the API) to be used for different types of file systems
  - the API is to the VFS interface, rather than any specific type of file system.

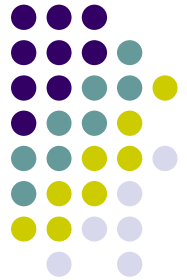
Schematic View of Virtual File System



Linux VFS

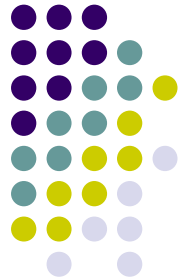

Virtual File System (VFS)
FAT
NTFS
ext3
...
reiser
/proc
Buffer Cache
Device Drivers
Physical Devices

Question



- Which of the following is incorrect about VFS?
  A. VFS allows an OS to support many different file systems
  B. VFS provides the same API for all file systems
  C. VFS is available in all OSes
  D. VFS hides the detailed implementation of each file system from programmers

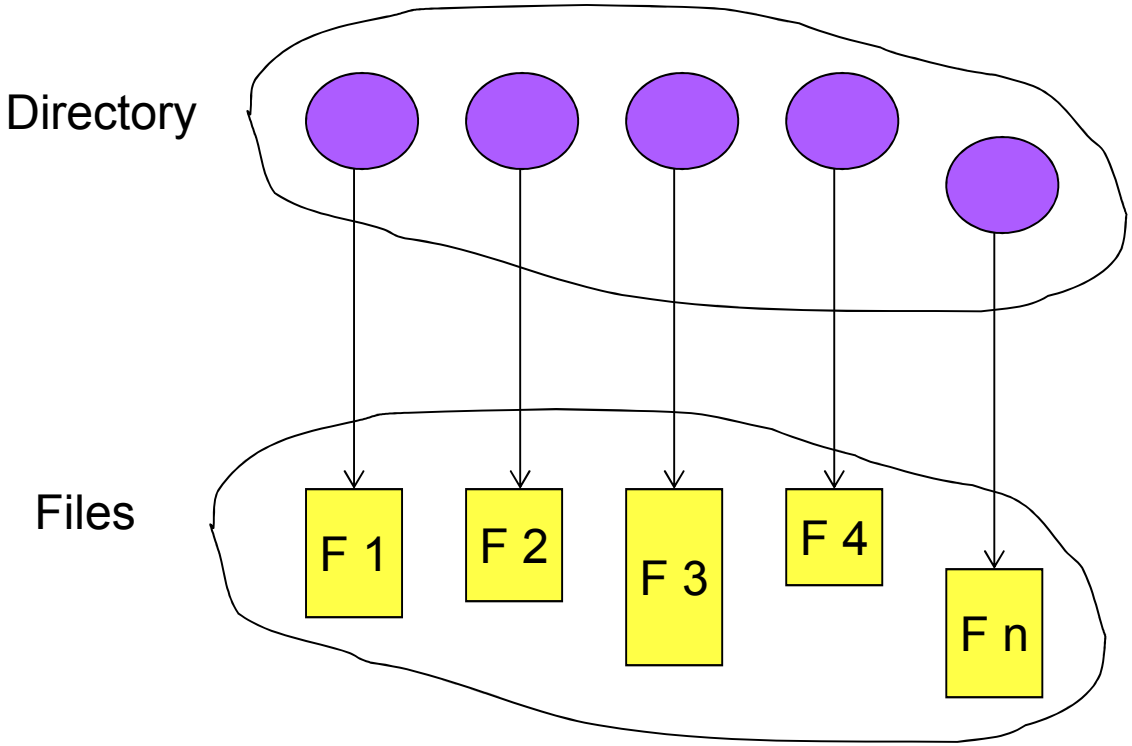
Question



- Which of the following is a correct description of a directory?
  A. a directory is a disk partition to contain files
  B. a directory is actually a file containing partial information about its files
  C. a directory is a container of files' data
  D. a directory contains the FCB and data of files

Directory Structure

- A collection of nodes containing information about all files



Directory Implementation

- **Linear list**
  - list of file names with pointer to the data blocks
  - simple to program
  - time-consuming to execute
  - FAT http://en.wikipedia.org/wiki/File_Allocation_Table
  - Linux FS (Ext3)
- **Hash Table**
  - linear list with hash data structure
  - decreases directory search time
  - **collisions** resolution
  - fixed size

Directory Implementation

- **Balanced binary tree**
  - RAISERFS http://en.wikipedia.org/wiki/ReiserFS

Directory in Linux

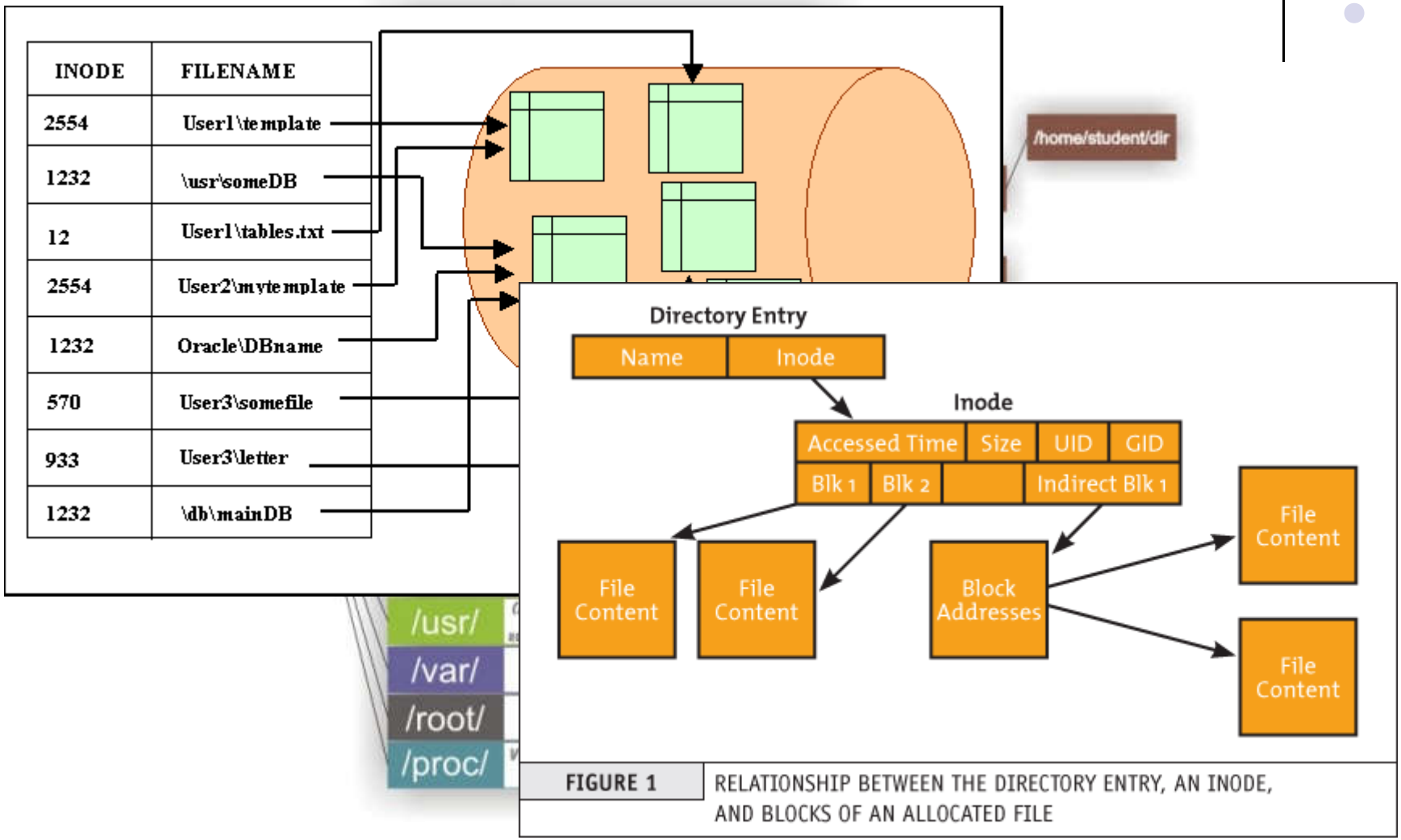
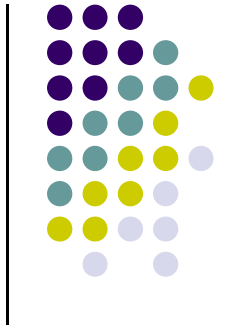
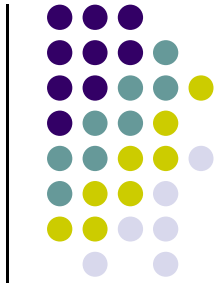


FIGURE 1 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN ALLOCATED FILE



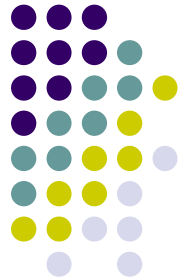
File management API

File Operations

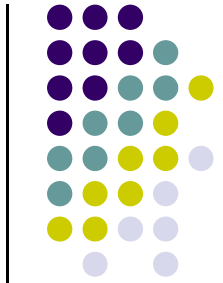


- Create
- Write
- Read
- Reposition within file
- Delete
- Truncate
- *Open(F_i)*
  - search the directory structure on disk for entry F_i, and move the content of entry to memory
- *Close (F_i)*
  - move the content of entry F_i in memory to directory structure on disk

Operations Performed on Directory

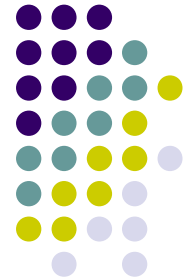


- Search for a file
- Create a directory
- Delete a directory
- List a directory
- Rename a directory
- Traverse the file system



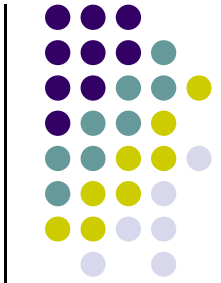
File/directory protection

Protection



- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

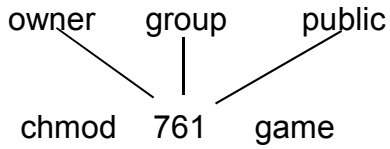
Access Lists and Groups



- Mode of access: read, write, execute
- Three classes of users

			RWX
a) **owner access**	7	⇒	1 1 1
b) **group access**	6	⇒	RWX 1 1 0
c) **public access**	1	⇒	RWX 0 0 1

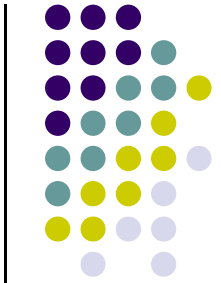
- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file

chgrp G game

Question



- Suppose a file has access mode 664. Which is correct ?

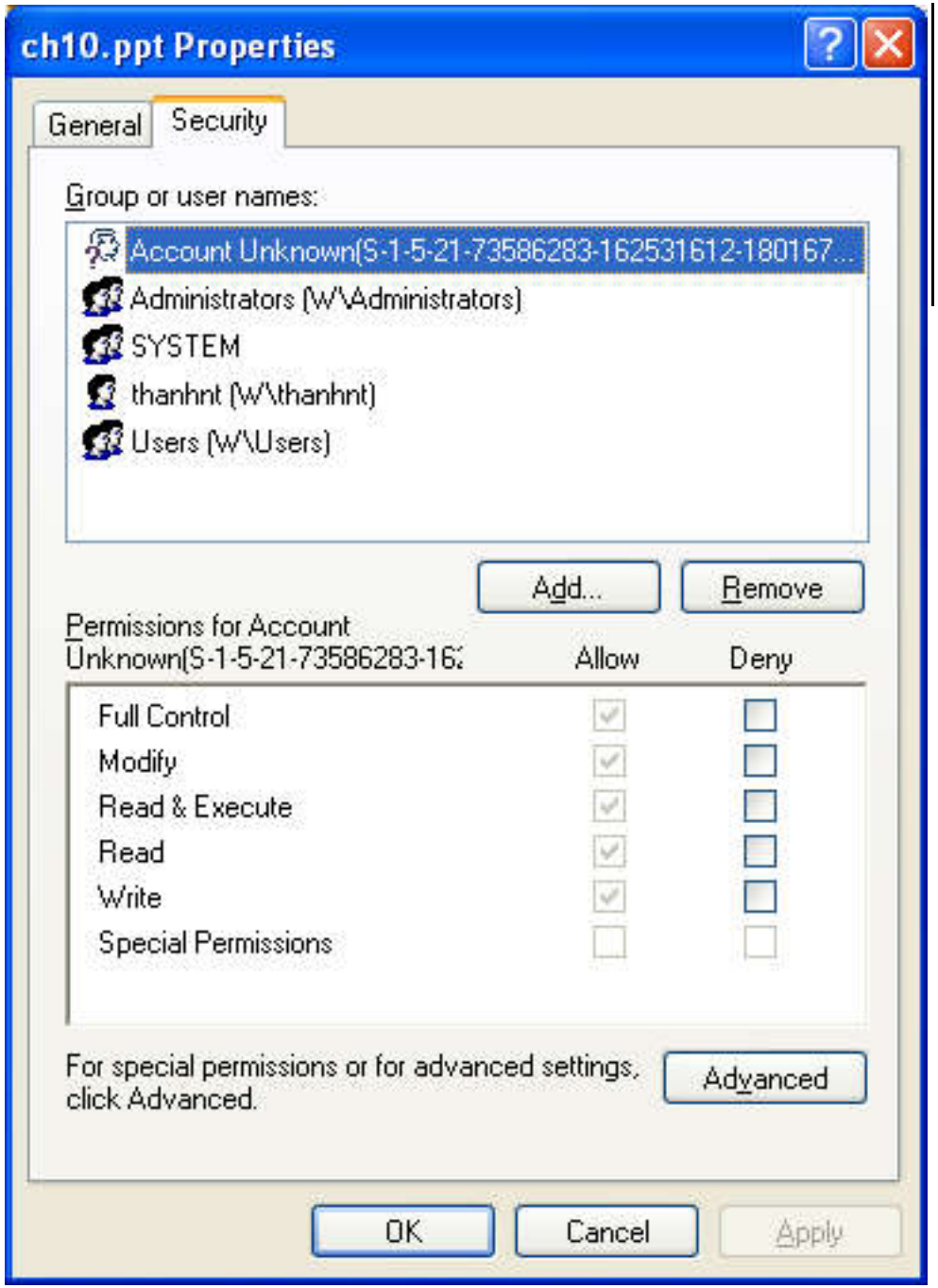
A. Any user can execute the file

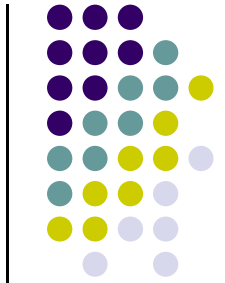
B. Users of the owner group can execute the file

C. Any user can read the file

D. The owner cannot write the file

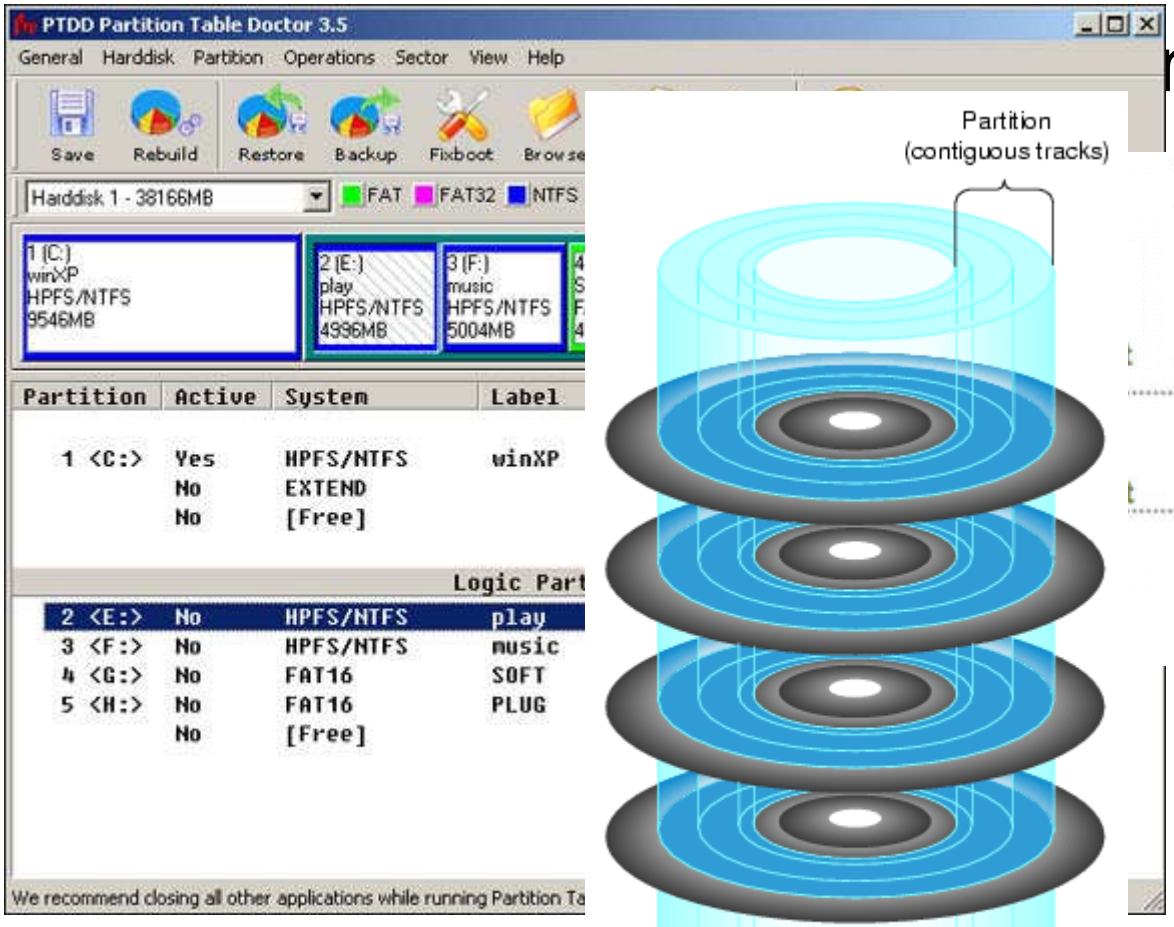
Windows XP Access-control List Management

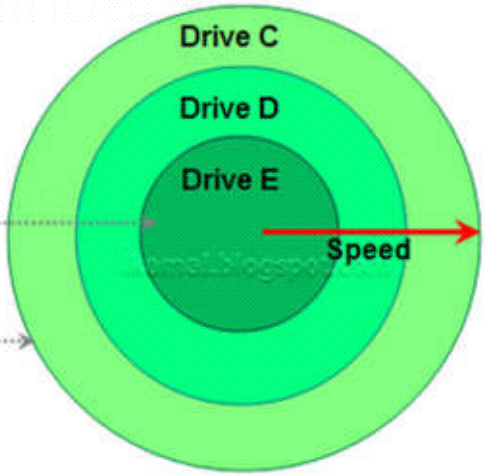




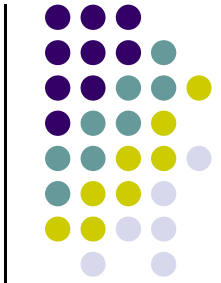
Storage Allocation Methods

Disk partitions

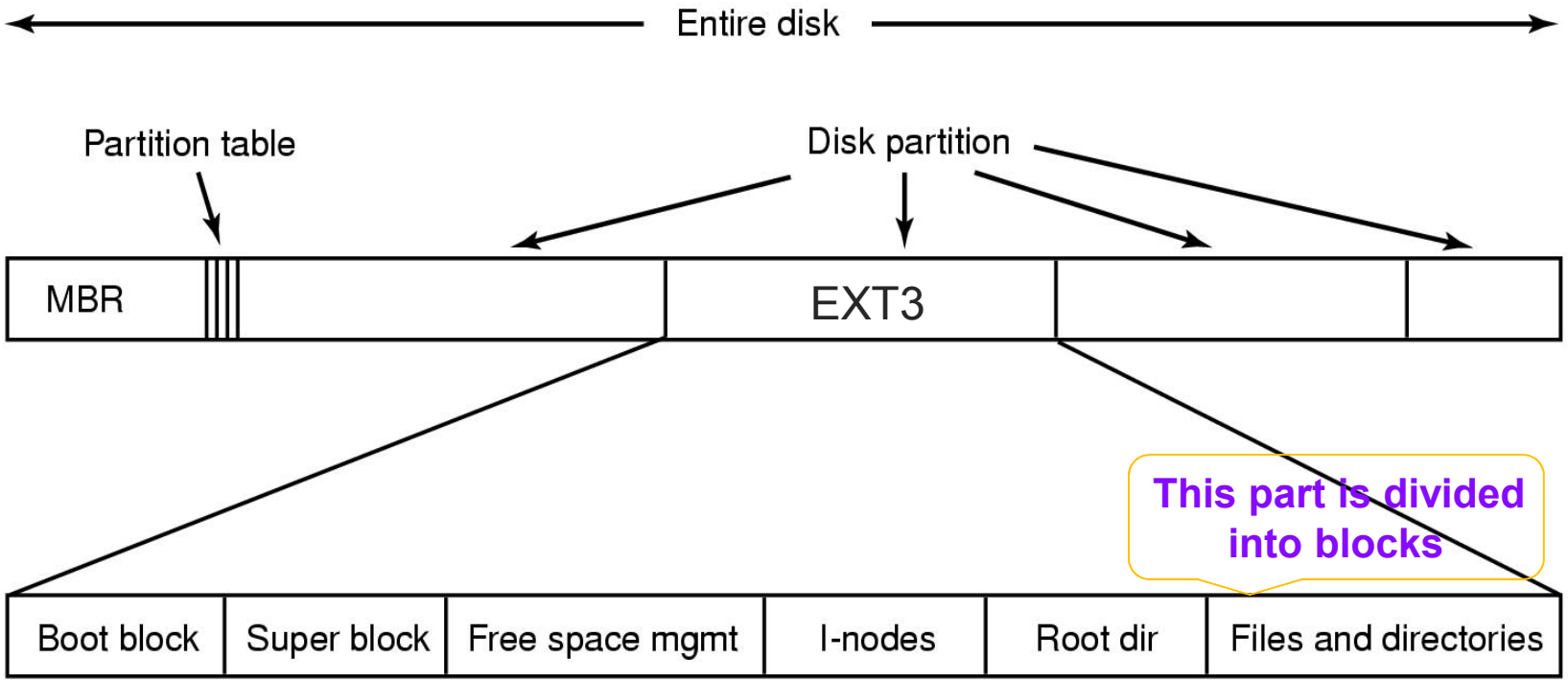




Partition organization



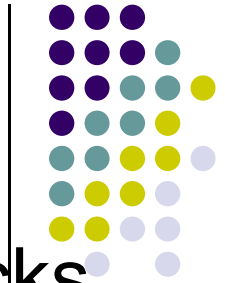
- Organization is specific to each OS
  - region for storing data is divided into equal **blocks**



Allocation Methods

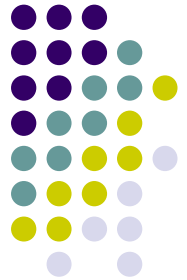
- An allocation method refers to how disk blocks are allocated for files
  - Contiguous allocation
  - Linked allocation
  - Indexed allocation
- Each method needs an appropriate way to access file

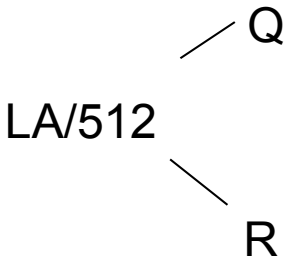
Contiguous Allocation



- Each file occupies a set of contiguous blocks on the disk
- Simple
  - only starting location (block #) and length (number of blocks) are required
- Random access
- Wasteful of space
  - dynamic storage-allocation problem
- Files cannot grow

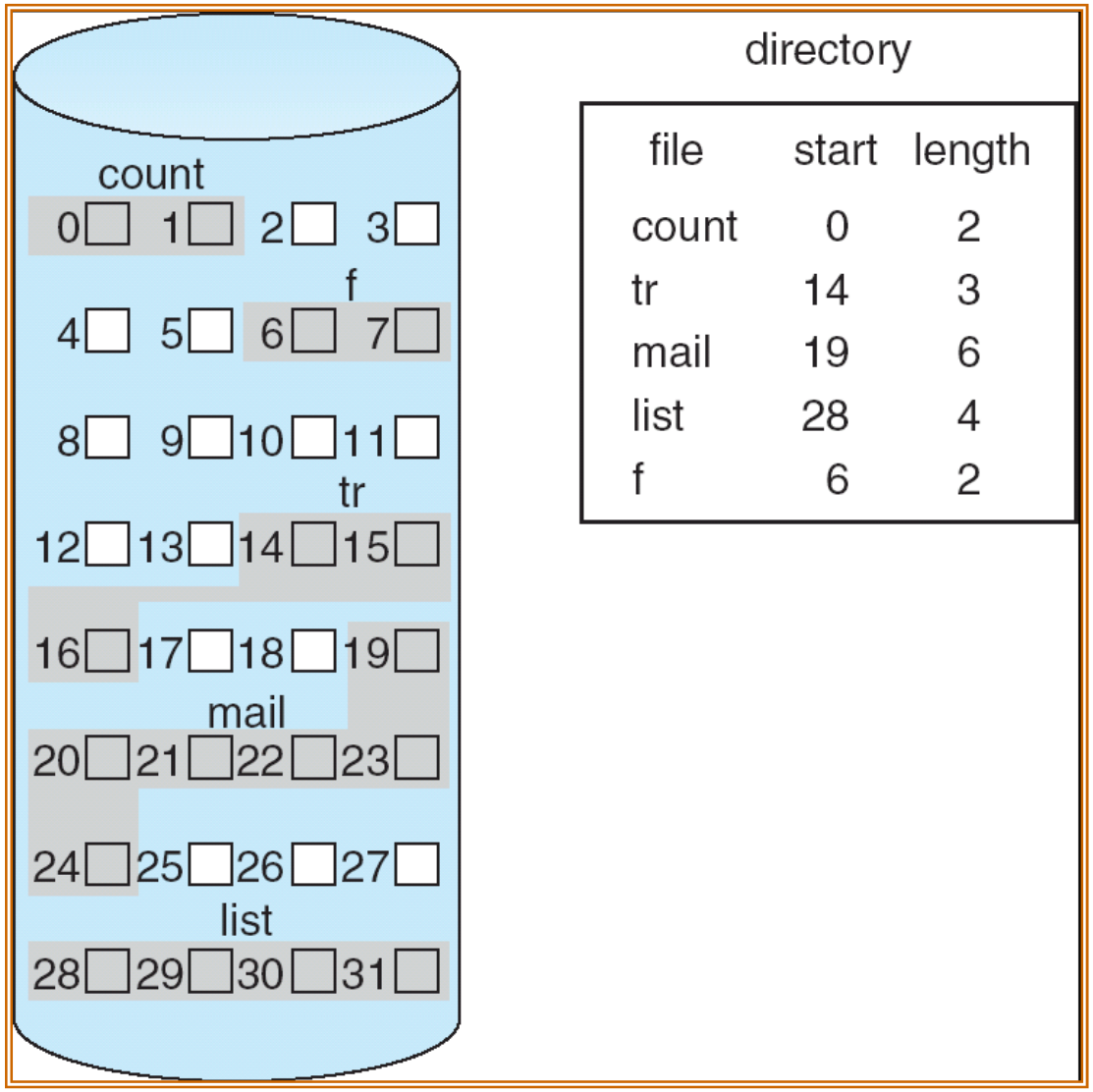
Contiguous Allocation



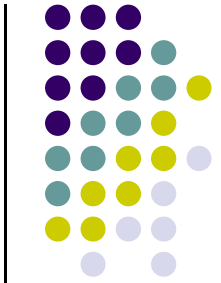
- Mapping from logical to physical
  - LA is position to access
  - 

Suppose block size is 512KB
Block to be accessed = Q + starting address
Displacement/offset into block = R

Contiguous Allocation of Disk Space



Question

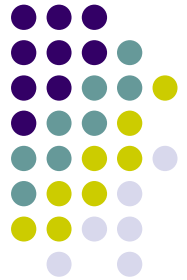


- A system using contiguous allocation
  - block size is 2KB
  - a file has the length of 12.3MB
- Which of the following is the correct location of file at position 50.5KB
  A. block 25, offset 512
  B. block 24, offset 1024
  C. block 25, offset 510
  D. block 24, offset 2512

Extent-Based Systems

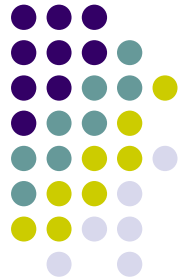
- Many newer file systems (I.e. Veritas File System)
  - use a modified contiguous allocation scheme
- Extent-based file systems
  - allocate disk blocks in **extents**
- An **extent** is a contiguous block of disks
  - Extents are allocated for file allocation
  - A file consists of one or more extents
  - extents of a file are not necessarily contiguous

Question



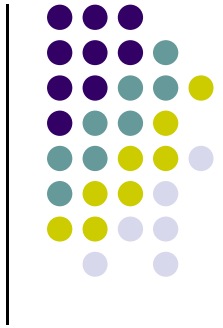
- An extent-based file system
  - an extent has *100* blocks
  - a block has size *2KB*
  - a file has size 25.3MB
- Which of the following is the correct extent number (started by 0) of file at position 15MB?
  A. 74
  B. 75
  C. 76
  D. 77

Question



- An extent-based file system
  - an extent has *100* blocks
  - a block has size *2KB*
  - a file has size 25.3MB
- Which of the following is the correct block (started by 0) number in the extent containing the data of file at position 15MB?
  A. 77
  B. 78
  C. 79
  D. 80

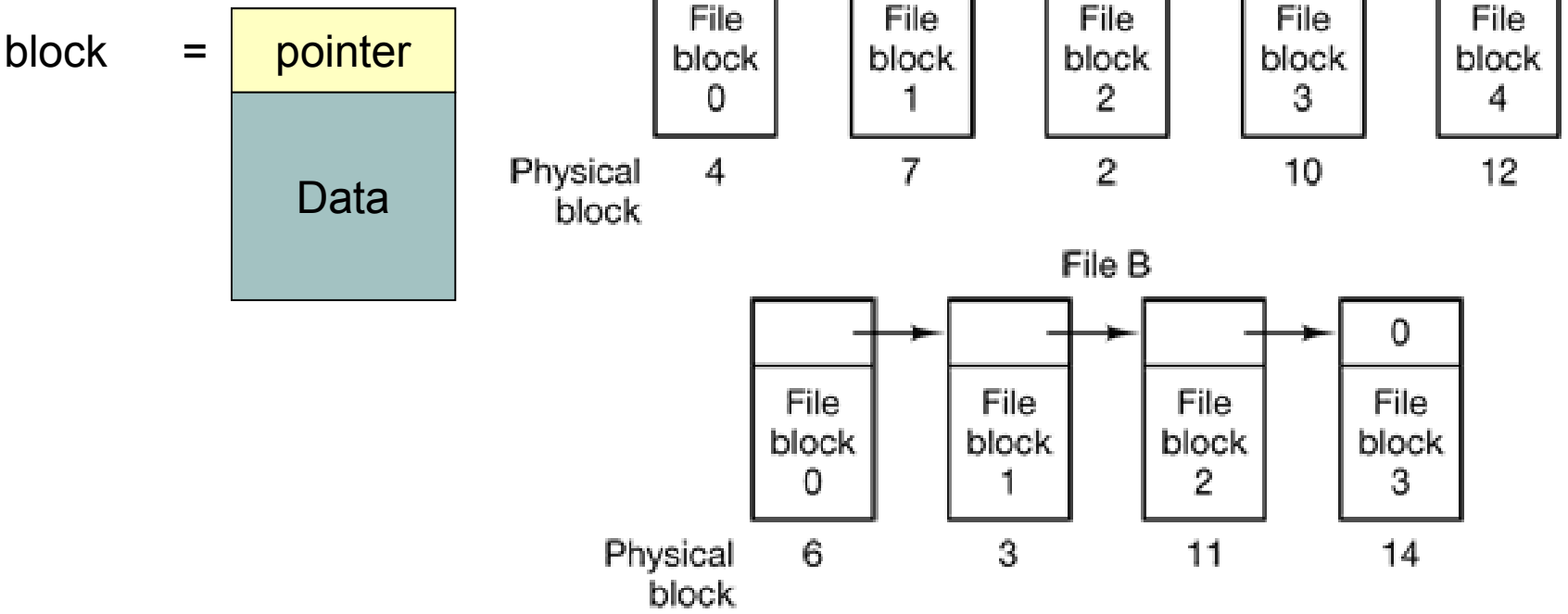
Question



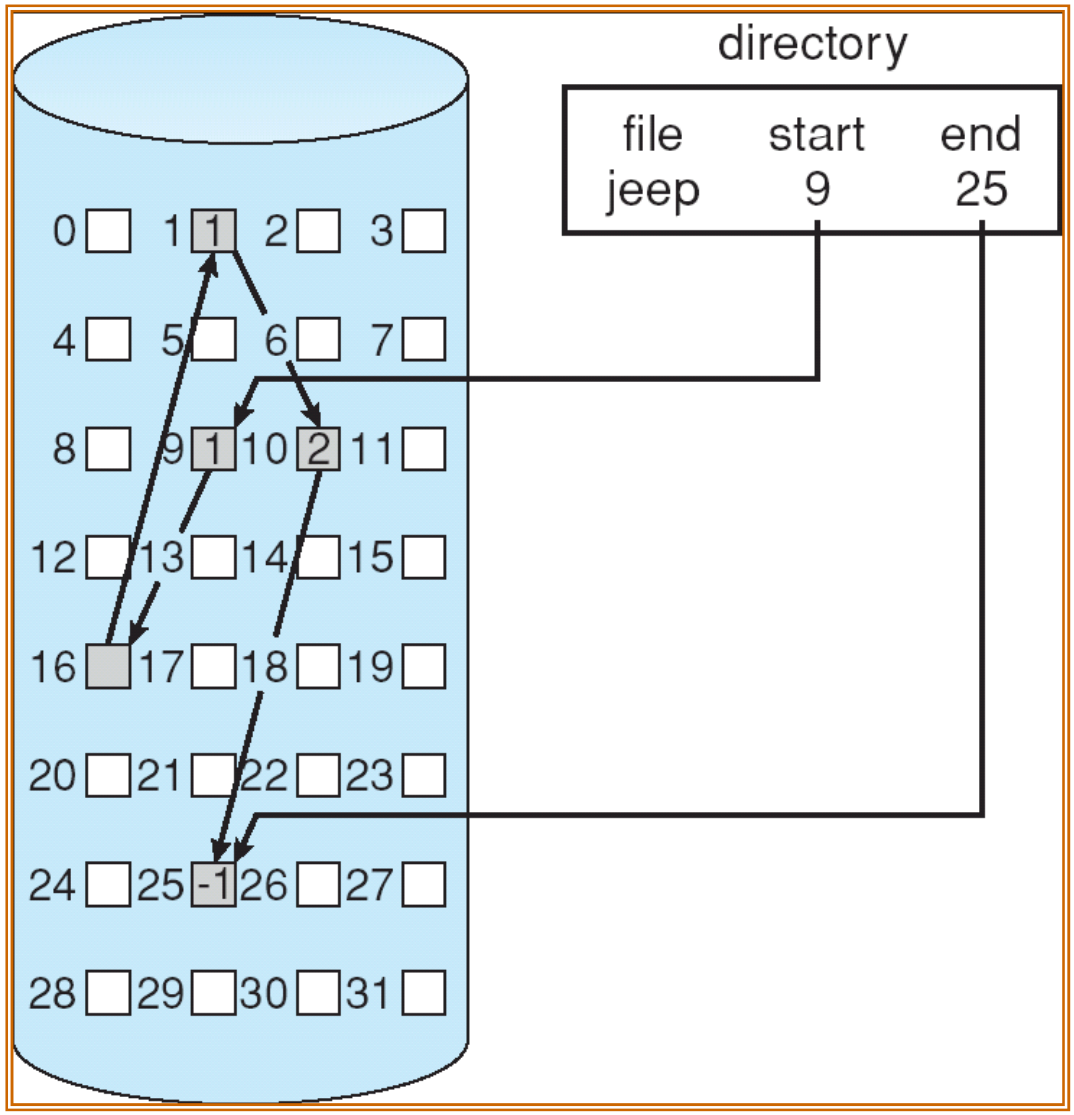
- An extent-based file system
  - an extent has *100* blocks
  - a block has size *2KB*
  - a file has size 25.3MB
- Which of the following is the correct offset of the of file at position 15MB in the block containing data?
  A. 0
  B. 2047
  C. 2048
  D. 512

Linked Allocation

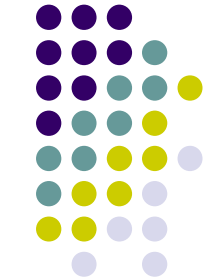
- Each file is a linked list of disk blocks:
  - blocks may be scattered anywhere on the disk.



Linked Allocation

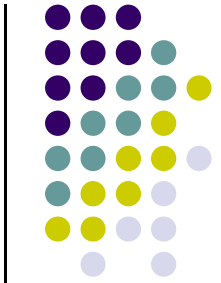


Question



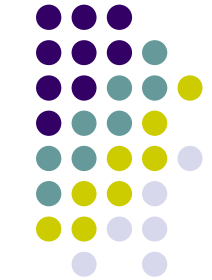
- A system uses linked list allocation
  - partition size 500GB (1GB=1024MB,1MB=1024KB,…)
  - block size *1*KB
- Which of the following is the correct size of the pointer of each block*?*
  A. short (2 bytes)
  B. int (4 bytes)
  C. float (4 bytes)
  D. long (8 bytes)

Question



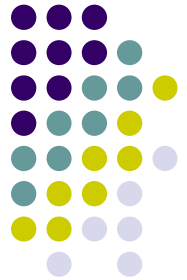
- A system uses linked list allocation
- Which of the following is the correct reason for no direct access*?*
  - A. because we don't know the location of block *n* directly
  - B. because data blocks of a file may be scattered
  - C. because of security reason
  - D. because the information of data block location is hidden

Question



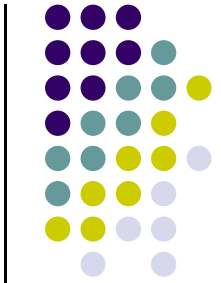
- A system uses linked list allocation
- Which of the following is the correct way to access block *n* of a file*?*
  - A. read the first block to find the location of block *n*
  - B. look up the location of block *n* from a table
  - C. recursively read block *n*-1 to find out the location of block *n*
  - D. there is no way to find the location of block *n*

Linked Allocation (Cont.)



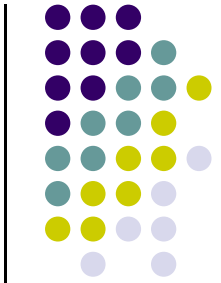
- Simple
  - need only starting address
- Free-space management system
  - no waste of space
- No random access
  - File-allocation table (FAT) – disk-space allocation used by MS-DOS and OS/2.

Question



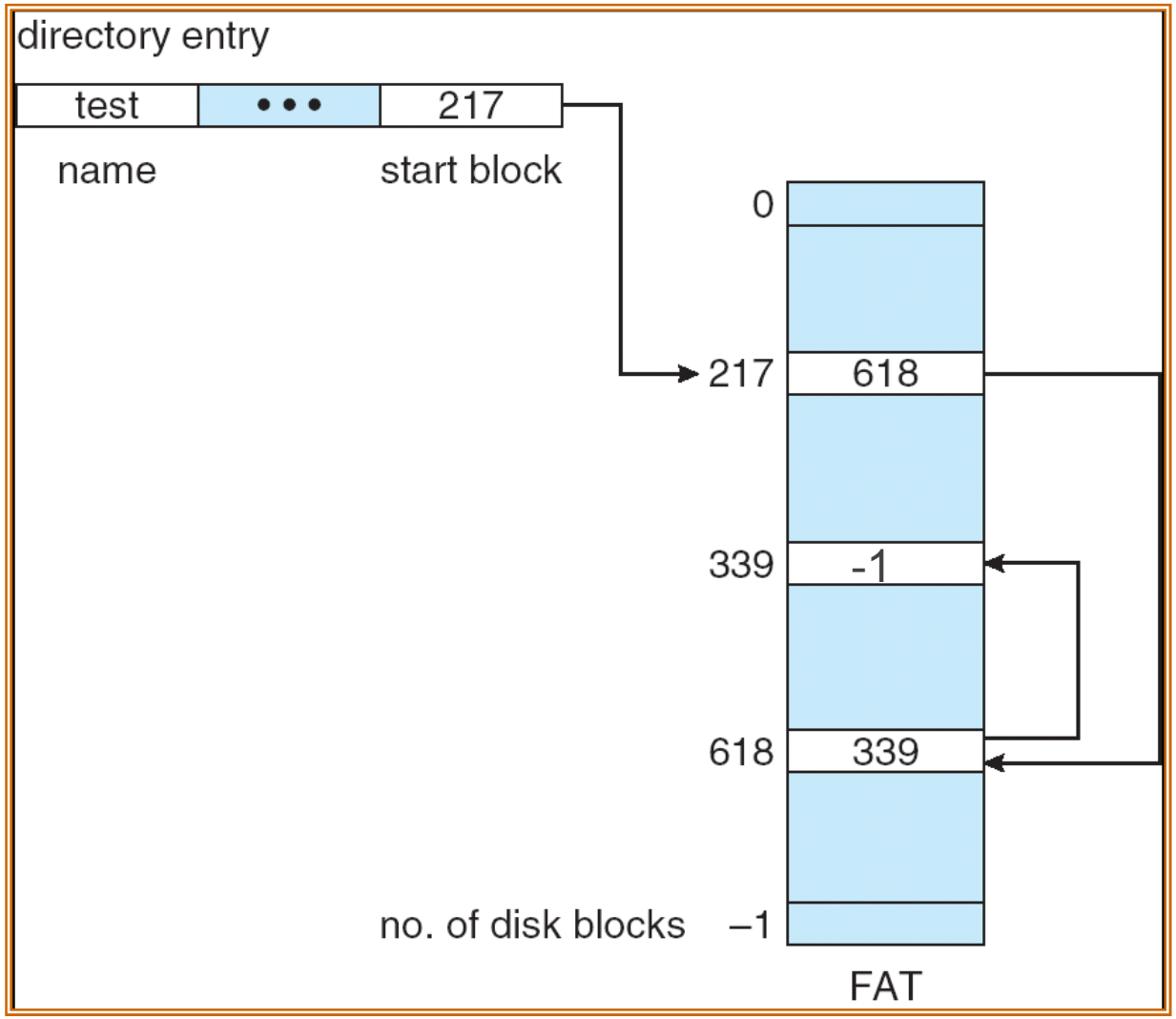
- A system uses linked list allocation
  - block size 2KB
  - pointer size 4 bytes
  - file size 15.4MB
- Which of the following is the correct block of file at position 15.25KB?
  A. 7
  B. 8
  C. 9
  D. 10

Question



- A system uses linked list allocation
  - block size 2KB
  - pointer size 4 bytes
  - file size 15.4MB
- Which of the following is the correct offset in the block of file at position 15.25KB?
  A. 1311
  B. 1312
  C. 1313
  D. 1314

File-Allocation Table (DOS)

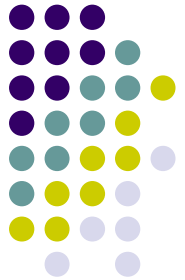


- Separate the pointers from data
- pointers are stored in File Allocation Table (FAT)
- The system stores two identical copies of FAT

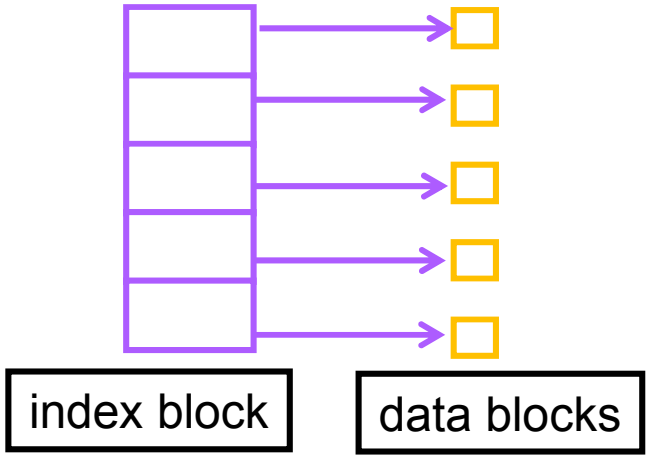
Question

- Which of the following is incorrect about FAT?
  A. it is fast to access the block *n* of a file
  B. it is slow to access the block *n* of a file
  C. if FAT is corrupted the whole partition is corrupted
  D. the system keeps two copies of FAT in order to reduce the risk of FAT corruption

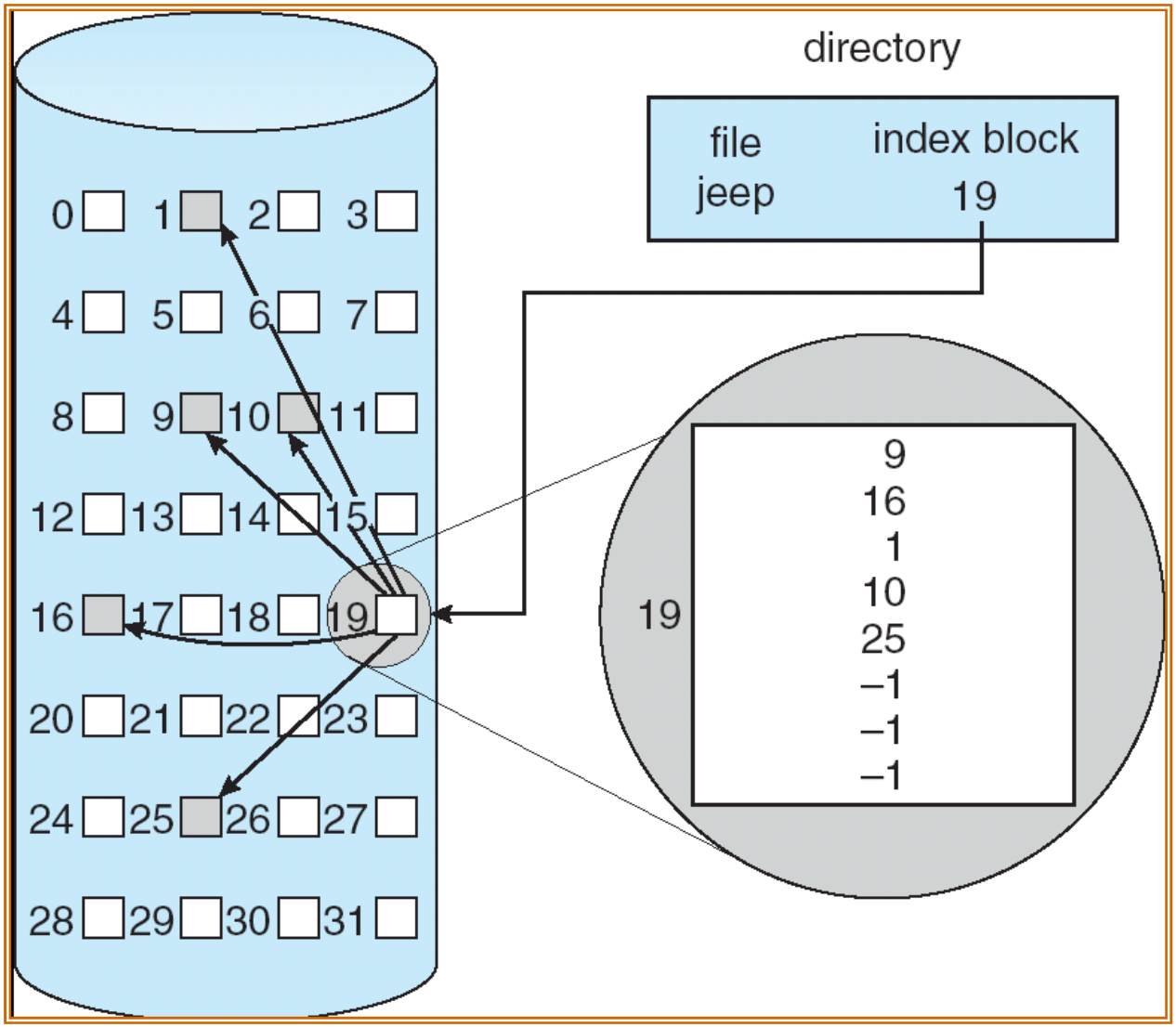
Indexed Allocation



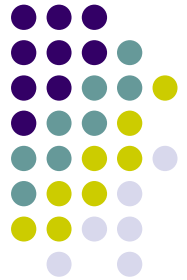
- Brings all pointers together into the *index block*
- Logical view



Example of Indexed Allocation

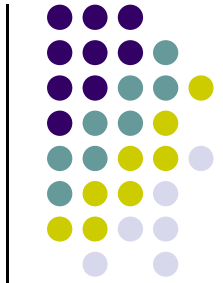


Indexed Allocation (Cont.)



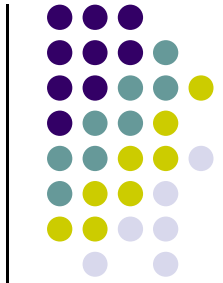
- Need index block
- Random access
- Dynamic allocation without external fragmentation
  - have overhead of index block

Question



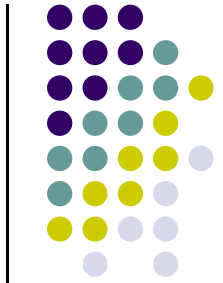
- System uses indexed allocation
  - block size 4KB
  - pointer size 4 bytes
- Which of the following is the correctly maximum file size?
  A. 4MB
  B. 8MB
  C. 16MB
  D. 32MB

Question

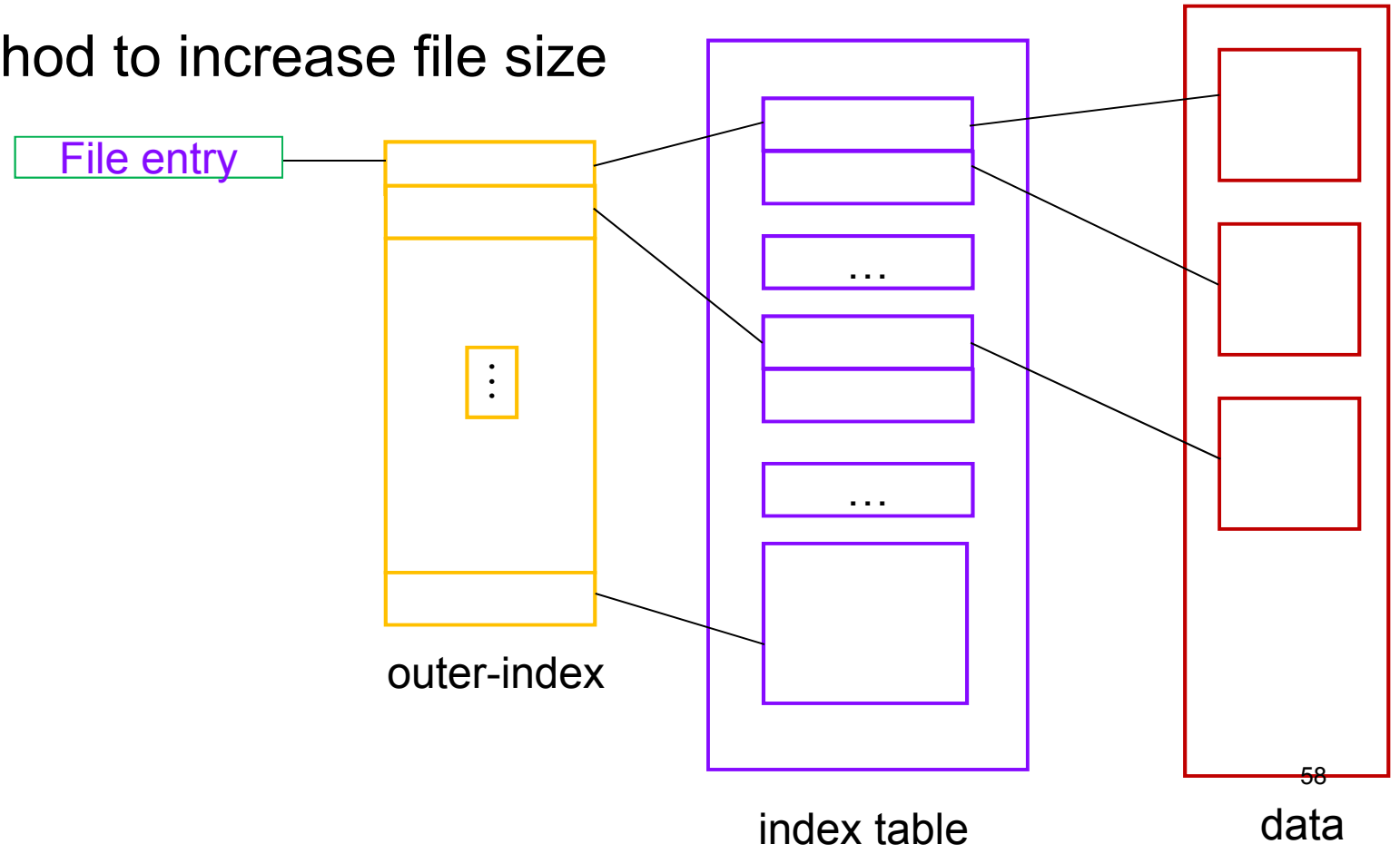


- System uses indexed allocation
  - block size 4KB
  - pointer size 4 bytes
  - file size 3MB
- Which of the following is the correct block (starting from 0) and offset at file position 35KB?
  A. (*block*, *offset*)=(9, 3071)
  B. (*block*, *offset*)=(9, 3070)
  C. (*block*, *offset*)=(8, 3072)
  D. (*block*, *offset*)=(8, 3070)

Indexed Allocation (cont'd)



- Two-level index
  - Two level of index block
  - Method to increase file size



Address locating

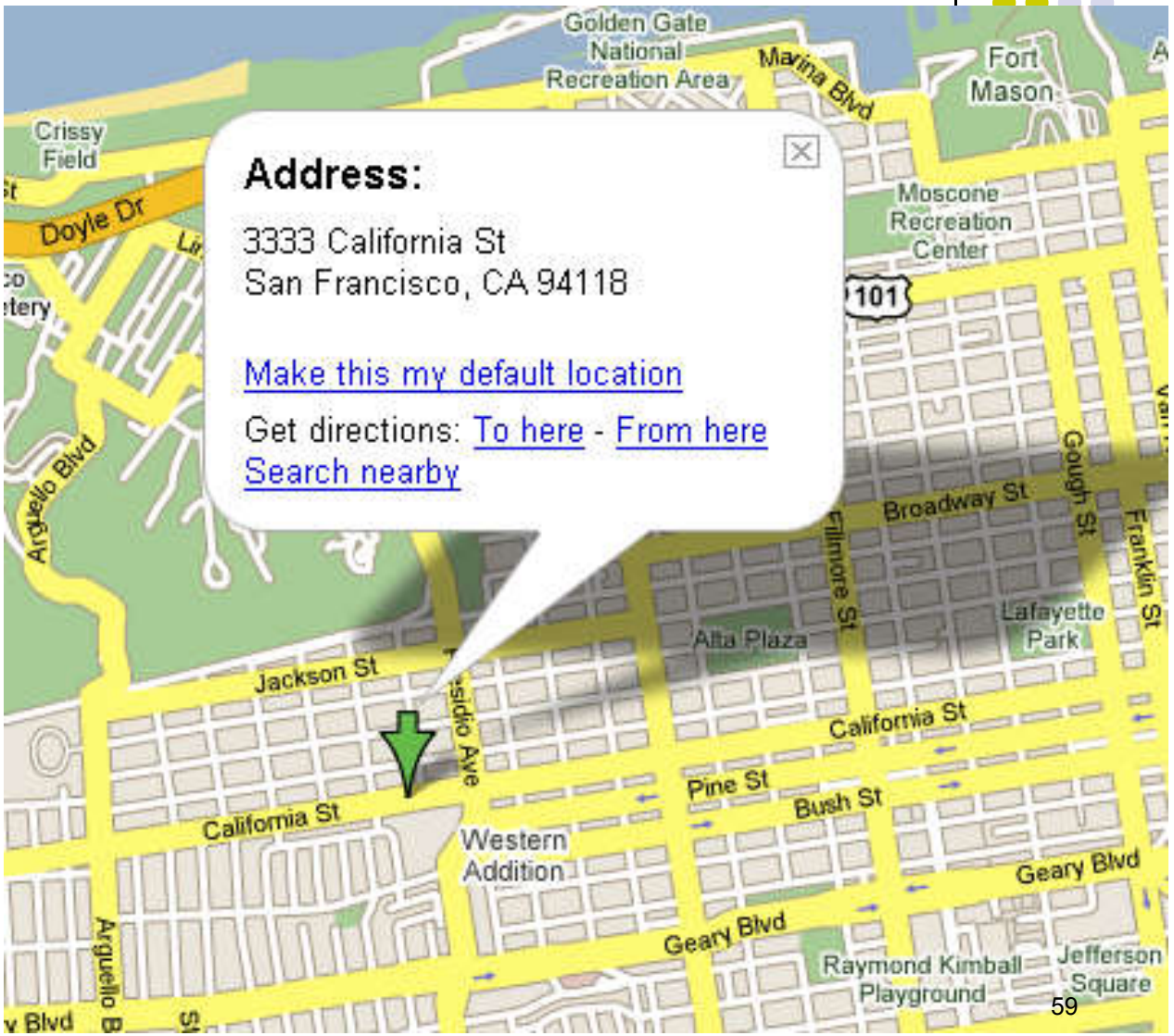
Go to USA →

San Francisco

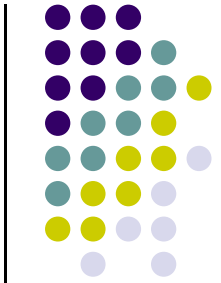
→

California St

→ 3333



Indexed Allocation (cont'd)



- A system uses two-level index
  - block size 4KB
  - pointer size 4 Bytes
- Which of the following is the correct maximum file size?
  A. 4GB
  B. 1GB
  C. 8GB
  D. 2GB

Indexed Allocation (cont'd)

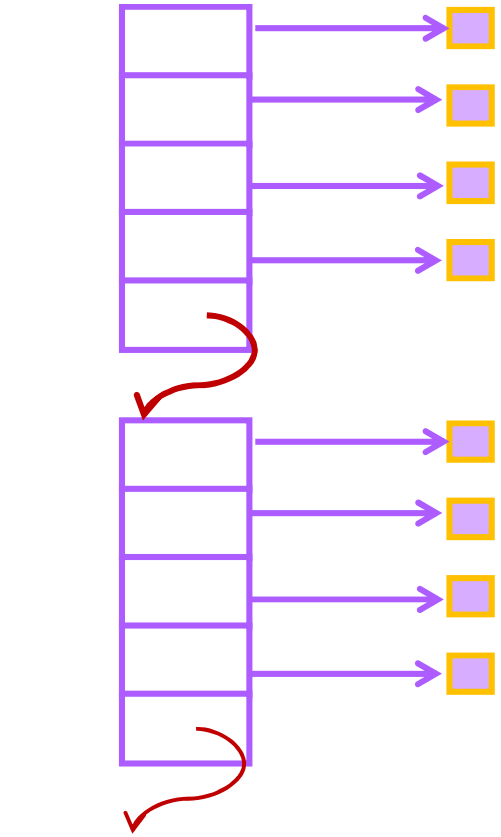
- A system uses two-level index
- Which of the following is the correct steps to locate the data at file position *n*?
  A. Identify block number $\rightarrow$ block number in block table $\rightarrow$ offset
  B. Identify block number in outer index block $\rightarrow$ block number $\rightarrow$ offset
  C. Identify offset $\rightarrow$ block number
  D. none of the above

Indexed Allocation (cont'd)

- A system uses two-level index
  - block size 4KB
  - pointer size 4 Bytes
  - File size 20MB
- Which is the correct address of file at position 15MB?
  A. (4,3072,0)
  B. (3,768,1023)
  C. (3,768,0)
  D. (3,3072,0)

Indexed Allocation (Cont.)

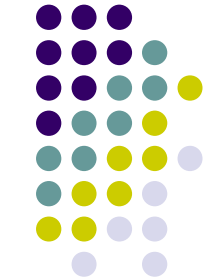
- Linked scheme
  - no limit on file size
- Combine linked list with index block
  - index blocks are linked
  - last pointer of a index block is the address of the next one



index table

data blocks

Question



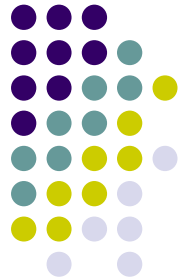
- Linked index block allocation
  - block size 2KB
  - pointer size 4 bytes
  - File size 20MB
- Which of the following is the correct steps to read file at position 15.5MB
  A. identify index block → offset → block number → read
  B. identify offset → block number → index block → read
  C. identify offset → index block → block number → read
  D. identify index block → block number →offset → read

Question

- Linked index block allocation
  - block size 2KB
  - pointer size 4 bytes
  - File size 20MB
- Which of the following is the correct index block number at file position 15.5MB (start from 0)

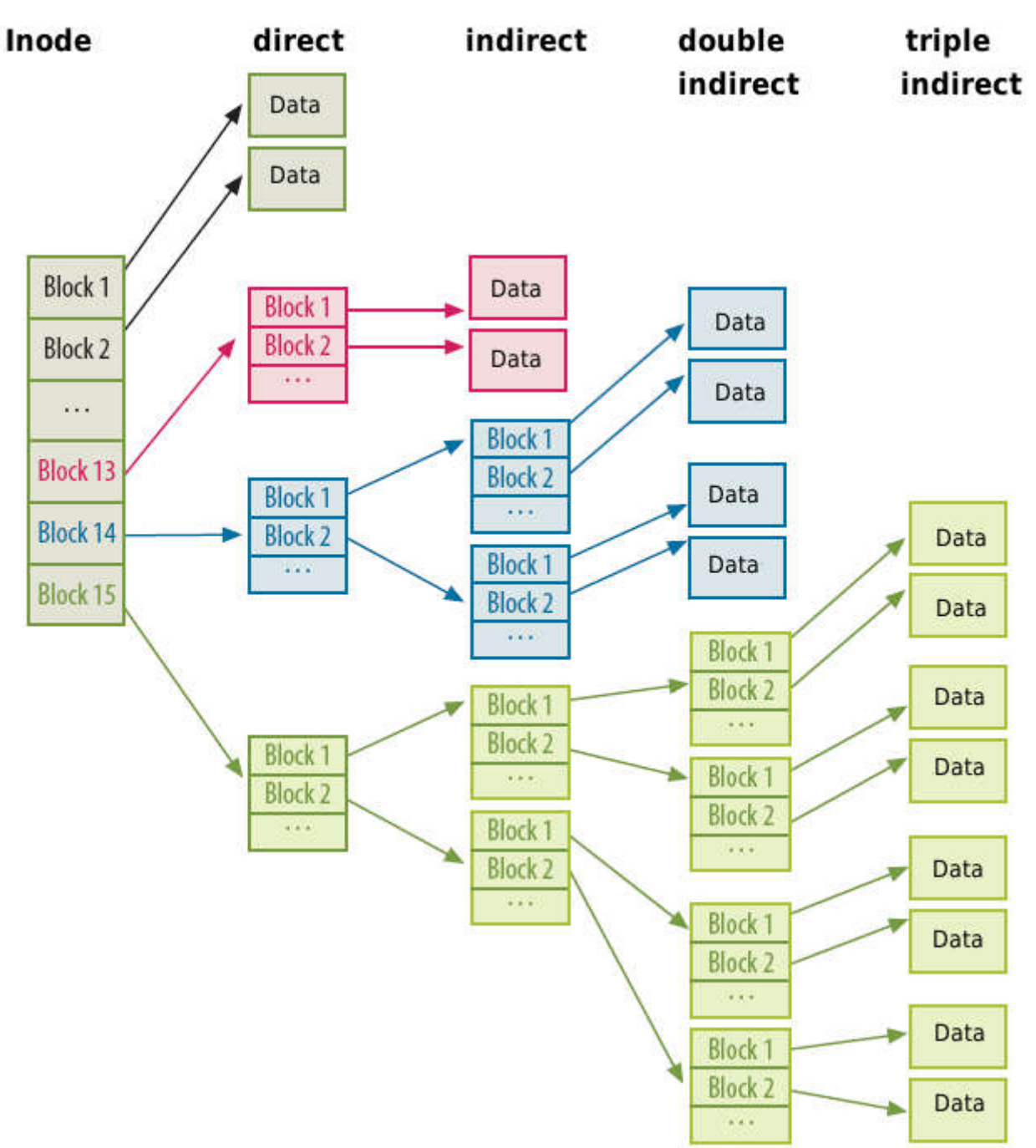
A. 13
B. 14
C. 15
D. 16

Question



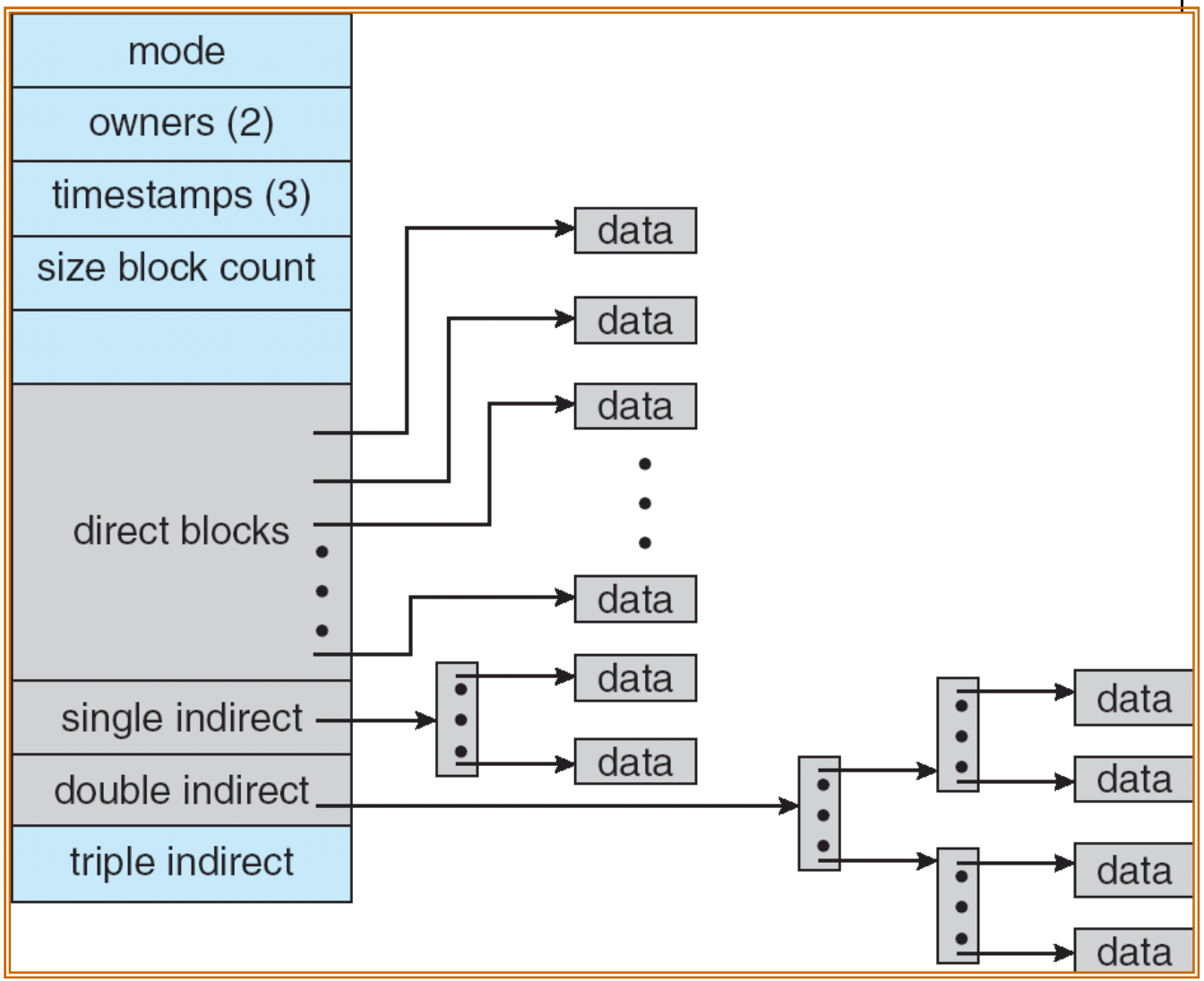
- Linked index block allocation
  - block size 2KB
  - pointer size 4 bytes
  - File size 20MB
- Which of the following is the correct block number and offset at file position 15.5MB
  A. (*block*, *offset*)=(271, 2047)
  B. (*block*, *offset*)=(271, 0)
  C. (*block*, *offset*)=(270, 2047)
  D. (*block*, *offset*)=(270, 0)

Combined Scheme: UNIX (4K bytes per block)

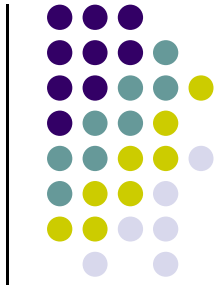


LINUX allocation

(10 direct pointers)

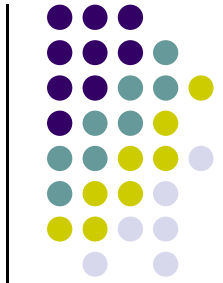


Question



- A UNIX system
  - pointer size *4* bytes
  - block size *4* KB
  - 12 direct pointers, 1 single indirect, 1 double indirect, 1 triple indirect pointers
- Which of the following is the correct maximum file size?
  A. $(12+2^{10}+2^{20}+2^{30})$KB
  B. $4*(2^{10}+2^{20}+2^{30})$KB
  C. 2^{32}KB
  D. $4*(12+2^{10}+2^{20}+2^{30})$KB

Question

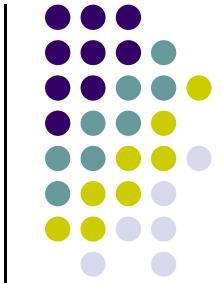


- A UNIX system
  - pointer size *4* bytes
  - block size *4* KB
  - 12 direct pointer, 1 single indirect, 1 double indirect, 1 triple indirect pointers
- Which of the following is the correct maximum number of indexed blocks?
  A. $(1+2^{10}+2^{20})$
  B. $(2+2^{10}+2^{20})$
  C. $(3+2^{11}+2^{20})$
  D. 2^{20}

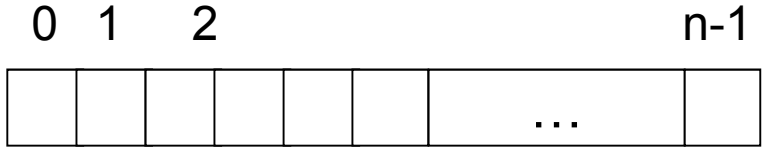
Question

- A UNIX system
  - pointer size *4* bytes
  - block size *4* KB
  - 12 direct pointer, 1 single indirect, 1 double indirect, 1 triple indirect pointers
  - File size 78MB
- Which of the following is the correct location at file position 95KB?
  A. triple indirect block, (*block*, *offset*)=(12,3071)
  B. double indirect block, (*block*, *offset*)=(12,3071)
  C. single indirect block, (*block*, *offset*)=(11,3071)
  D. single indirect block, (*block*, *offset*)=(11,3072)

Free-Space Management



- Bit vector (*n* blocks), e.g. Linux (ext3)
  - Easy to get contiguous blocks

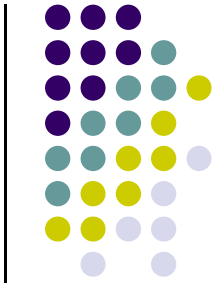


$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{block}[i] \text{ free} \\ 1 \Rightarrow \text{block}[i] \text{ occupied} \end{cases}$$

- Bit map requires extra space
  Example:

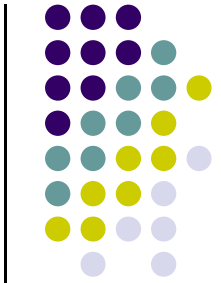
  block size = 2^{12} bytes
  disk size = 2^{30} bytes (1 gigabyte)
  $n = 2^{30}/2^{12} = 2^{18}$ bits (or 32K bytes)

Free-Space Management (Cont.)



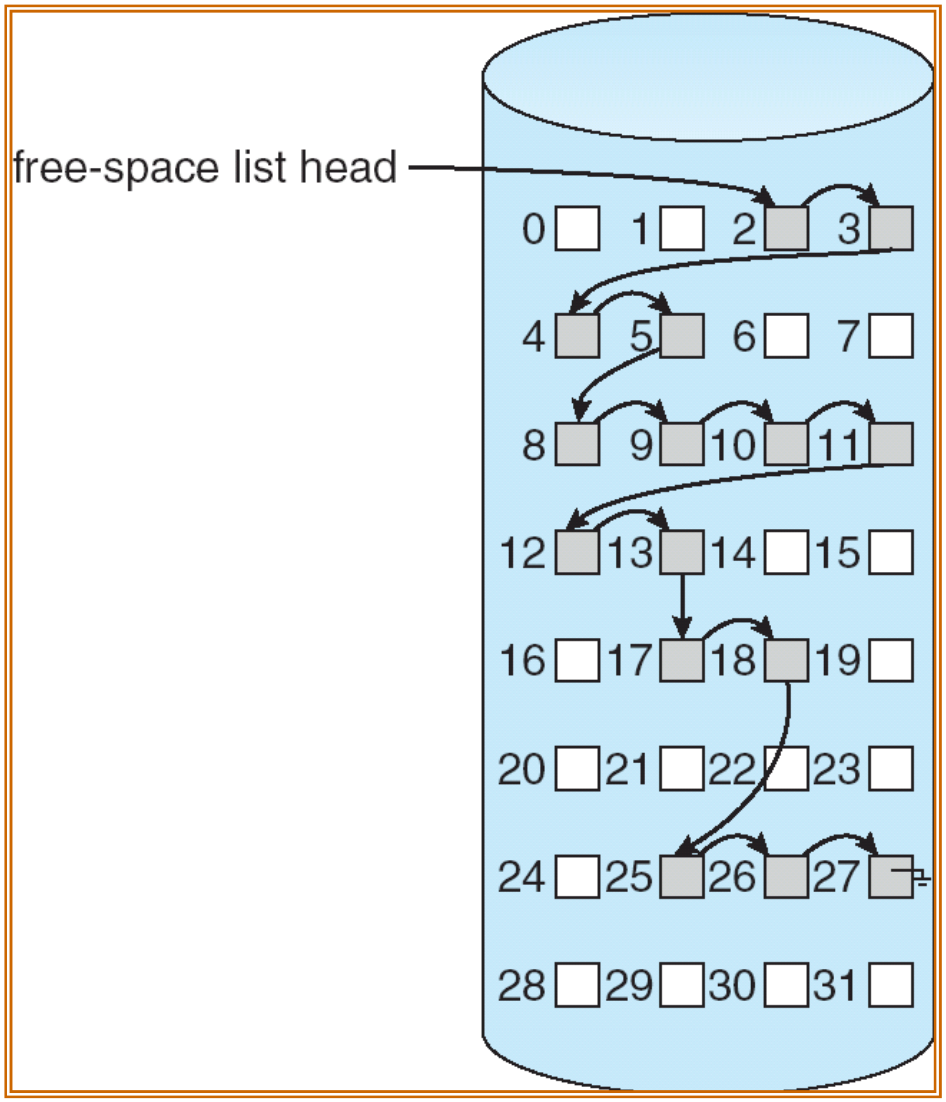
- Linked list (free list)
  - Cannot get contiguous space easily
  - No waste of space
- Grouping
  - Use linked blocks to store pointers to free blocks
  - Last pointer in a block is the address of the next one
- Counting
  - several contiguous blocks are freed/allocated for a file
  - each entry has
    - address of the first free block
    - number of contiguously free blocks

Free-Space Management (Cont.)



- Need to protect:
  - Pointer to free list
  - Bit map
    - Must be kept on disk
    - Copy in memory and disk may differ
    - Cannot allow for block[*i*] to have a situation where bit[*i*] = 1 in memory and bit[*i*] = 0 on disk
  - Solution:
    - Set bit[*i*] = 1 in disk
    - Allocate block[*i*]
    - Set bit[*i*] = 1 in memory

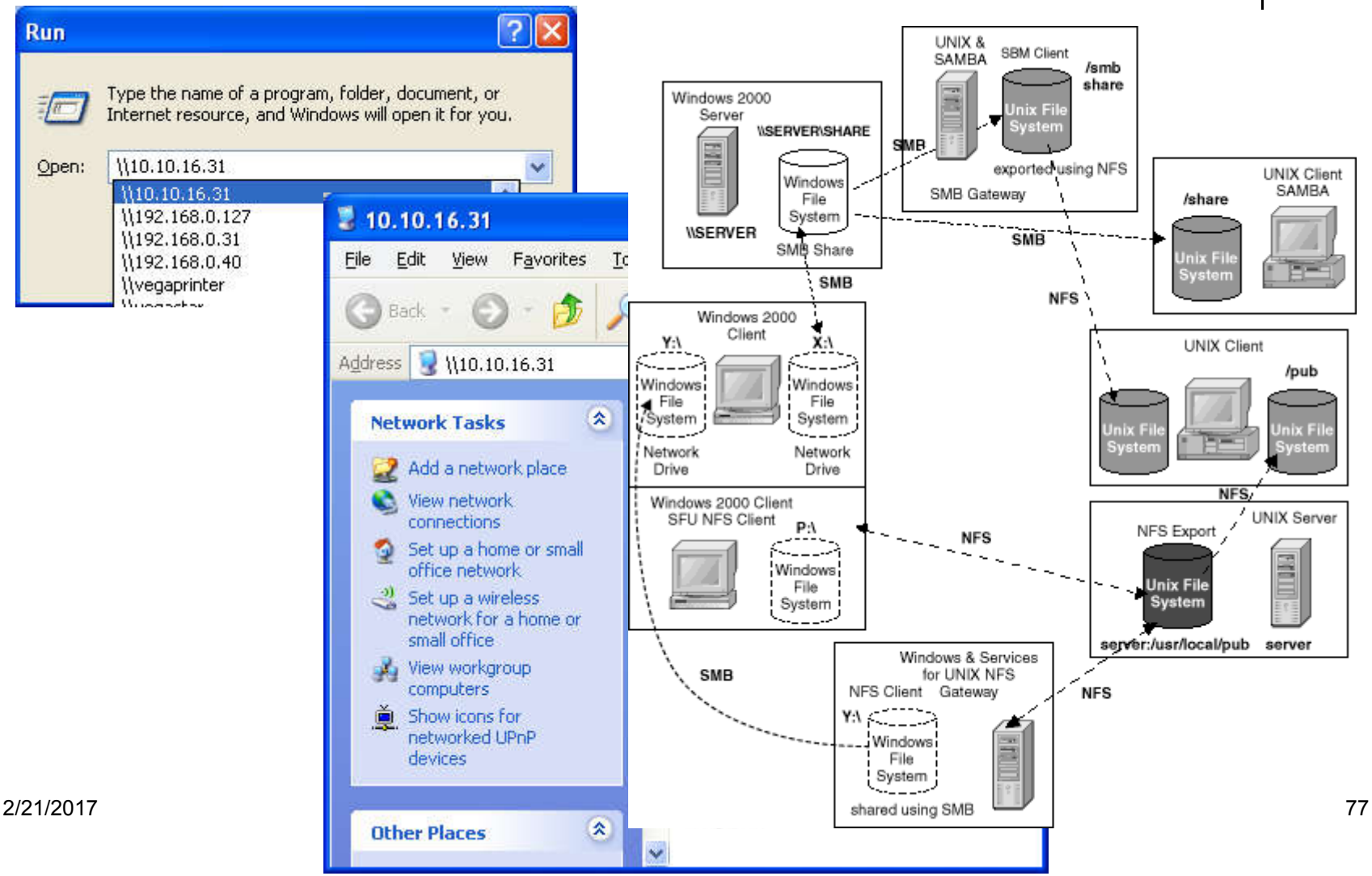
Linked Free Space List on Disk



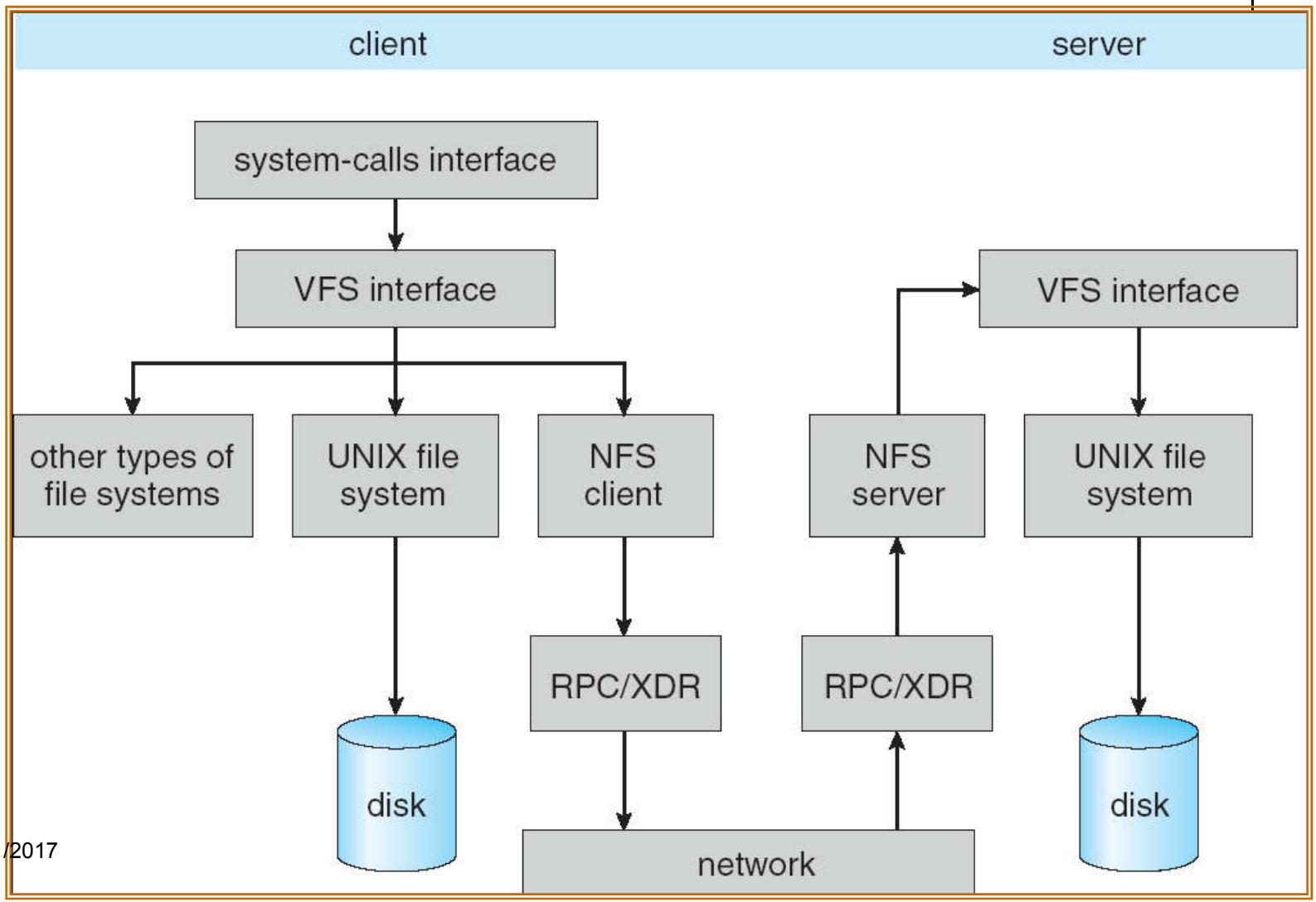
Shared file systems

- A file system is shared to other machines
  - the file system may be large and powerful
  - file sharing is needed in many applications
  - available in many systems, e.g., Windows, Linux (NFS)

Shared file systems



Network File System (NFS) Architecture



Question?