Files

Reading: Section 6.1-6.3 of Tanenbaum & Ch. 11& 12 from Silberschatz.

How to store the large amount of data into the computer?

What happens, when process terminates or killed using some data?

How to assign the same data to the multiple processes?

The solution to all these problems is to store information on disks or on other external media called files.

Files

A file is named collection of related information normally resides on a secondary storage device such as disk or tape.

Commonly, files represent programs (both source and object forms) and data; data files may be numeric, alphanumeric, or binary.

Information stored in files must be persistent,- not be effected by power failures and system reboot.

The files are managed by OS.

The part of OS that is responsible to manage files is known as the file system.

Files System Issues

How to *create* files?
How they *named*?
How they are *structured*?
What *operation* are allowed on files?
How to *protect* them?
How they *accessed* or *used*?
How to *implement*?

File Naming

When a process creates a file, it gives the file name; while process terminates, the file continue to exist and can be accesses by other processes.

A file is named, for the convenience of its human users, and it is referred to by its name. A name is string of characters.

The string may be of digits or special characters (eg. 2, !,% etc). Some system differentiate between the upper and lower case character, whereas other system consider the equivalent (like Unix and MS-DOS).

File Naming

Normally the string of max 8 characters are legal file name (e.g., in DOS), but many recent system support as long as 255 characters (eg. Windows 2000).

Many OSs support two-part file names; separated by period; the part following the period is called the file extension and usually indicates something about the file (e.g., file.c – C source file). But in some system it may have two or more extension such as in Unix proc.c.Z - C source file compressed using Ziv-Lampel algorithm.

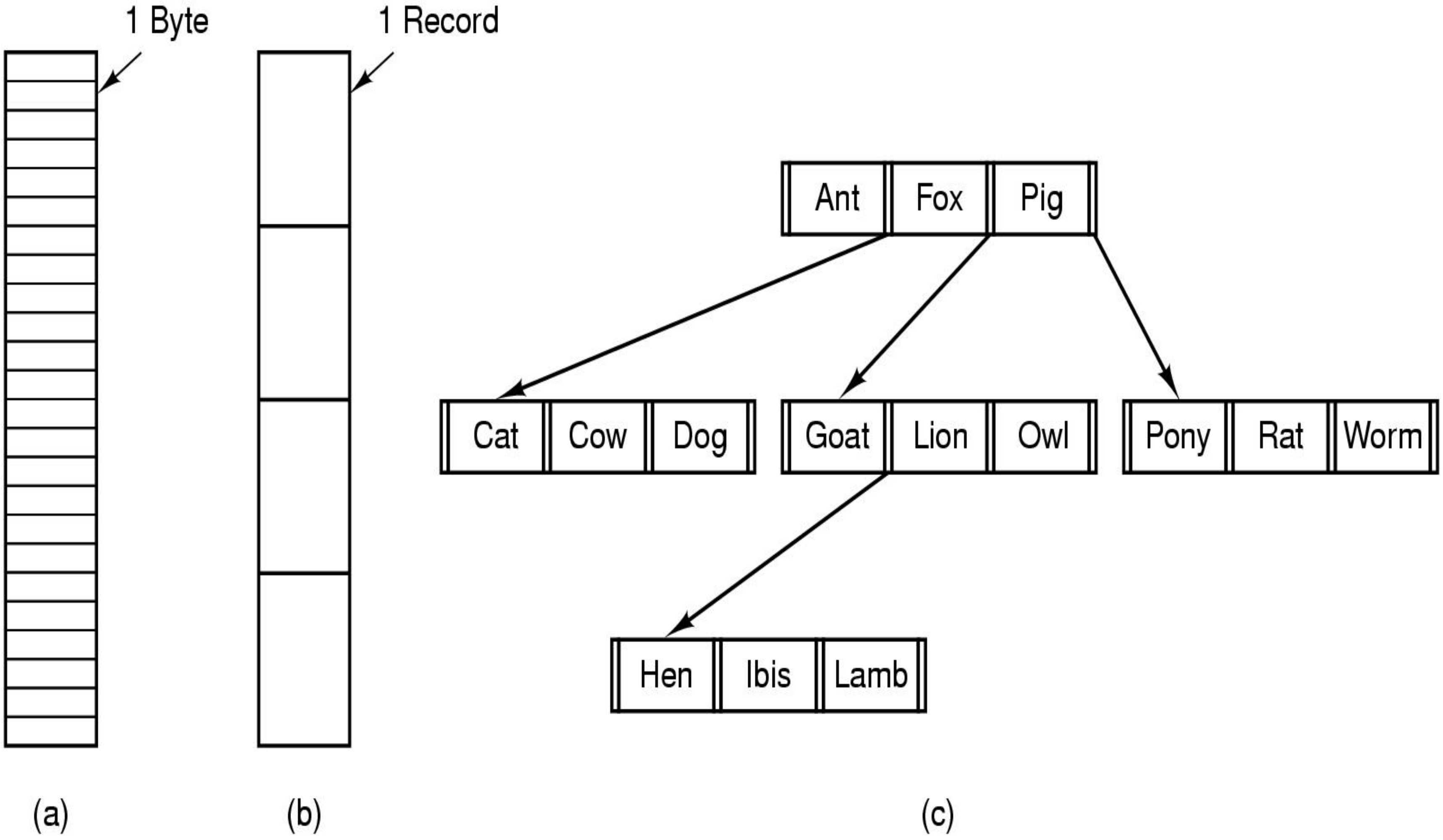
In some system (e.g., Unix), file extension are just conventions; in other system it requires (e.g., C compiler must requires .c source file).

File Structure

Files must have structure that is understood by OS. Files can be structured in several ways. The most common structures are:

Unstructured
Record Structured
Tree Structured

File Structure



a) Unstructured b) Record structured c) Tree structured

File Structure

Unstructured:

- Consist of unstructured sequence of bytes or words.
- OS does not know or care what is in the file.
- Any meaning must be imposed by user level programs.
- Provides maximum flexibility; user can put anything they want and name they anyway that is convenient.
- Both Unix and Windows use these approach.

File Structure

Record Structured:

- A file is a sequence of fixed-length records, each with some internal structure.
- Each read operation returns one records, and write operation overwrites or append one record.
- Many old mainframe systems use this structure.

Tree Structured:

- File consists of tree of records, not necessarily all the same length.
- Each containing a key field in a fixed position in the record, sorted on the key to allow the rapid searching.
- The operation is to get the record with the specific key.
- Used in large mainframe for commercial data processing.

File Types

Many OS supports several types of files

Regular files: contains user information, are generally *ASCII* or *binary*.

Directories: system files for maintaining the structure of file system.

Character Special files: related to I/O and used to model serial I/O devices such as terminals, printers, and networks.

Block special files: used to model disks.

File Types

ASCII files:

Consists of line of text.

Each line is terminated either by carriage return character or by line feed character or both.

They can be displayed and printed as is and can be edited with ordinary text editor.

Binary files:

Consists of sequence of byte only.

They have some internal structure known to programs that use them (e.g., executable or archive files).

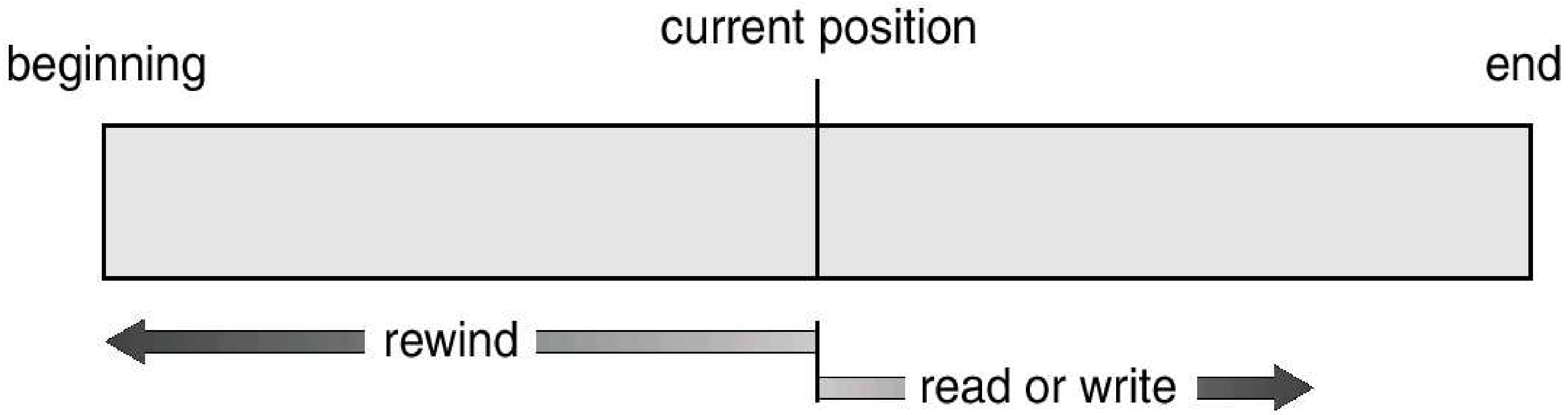
Many OS use extension to identify the file types; but Unix like OS use a magic number to identify file types.

Access Methods

Sequential and Direct access

Sequential Access:

The simplest excess method; Information in the file is processed in order, one record after the other.
Convenient when the storage medium is magnetic tap.
Used in many early systems.



Access Methods

Direct Access:

Files whose bytes or records can be read in any order. Based on disk model of file, since disks allow random access to any block.

Used for immediate access to large amounts of information.

When a query concerning a particular subject arrives, we compute which block contain the answer, and then read that block directly to provide desired information.

Operations: *read n, write n* (n is block number) or

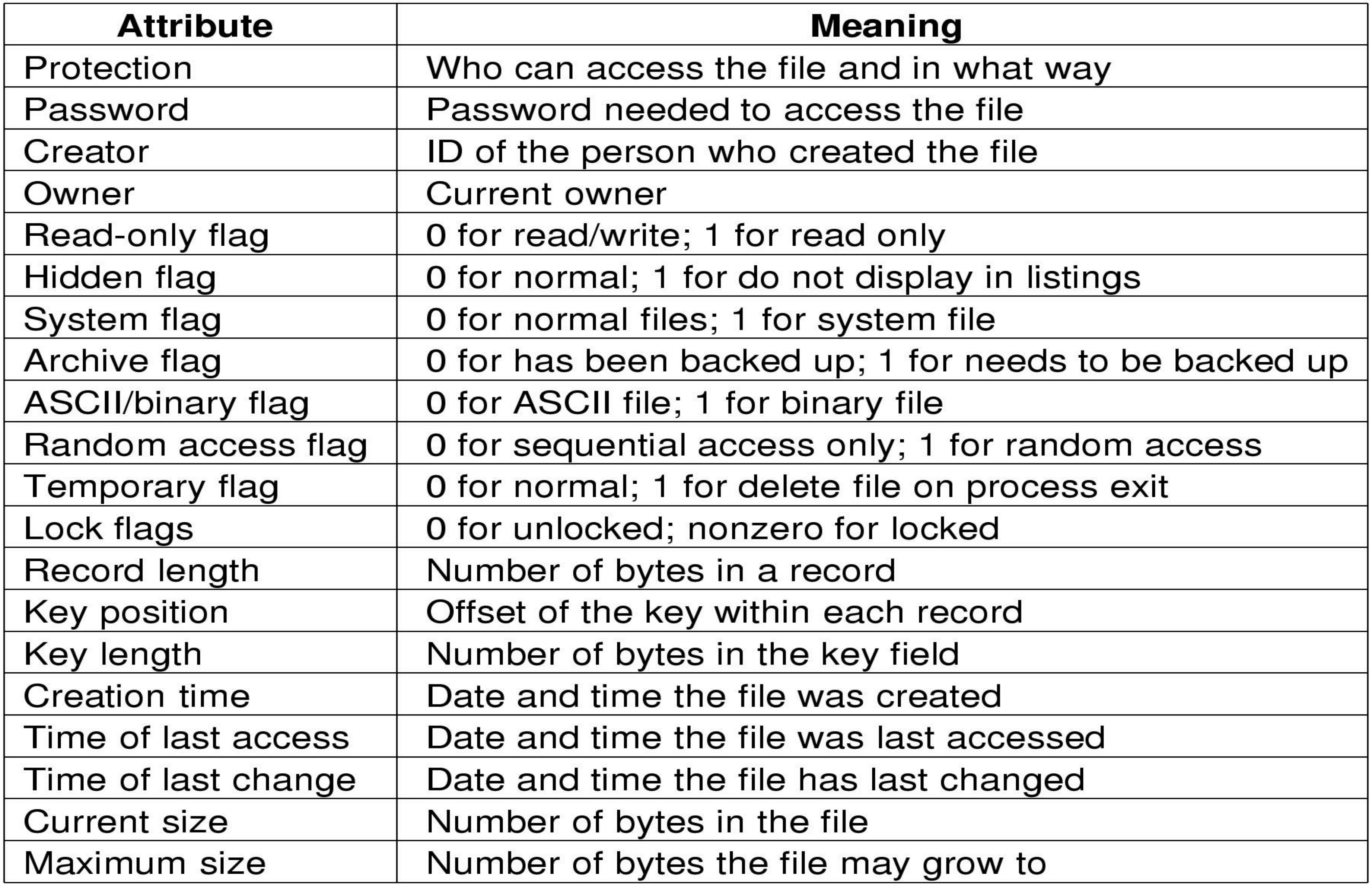
seek – to set current position. File can be access

sequentially from the current position.

File Attributes

In addition to name and data, all other information about file is termed as file attributes.
The file attributes may very from system to system.
Some common attributes are listed here.

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

File Operations

OS provides system calls to perform operations on files. Some common calls are:

Create: If disk space is available it create new file without data.

Delete: Deletes files to free up disk space.

Open: Before using a file, a process must open it.

Close: When all access are finished, the file should be closed to free up the internal table space.

Read: Reads data from file.

File Operations

Append: Adds data at the end of the file.

Seek: Repositions the file pointer to a specific place in the file.

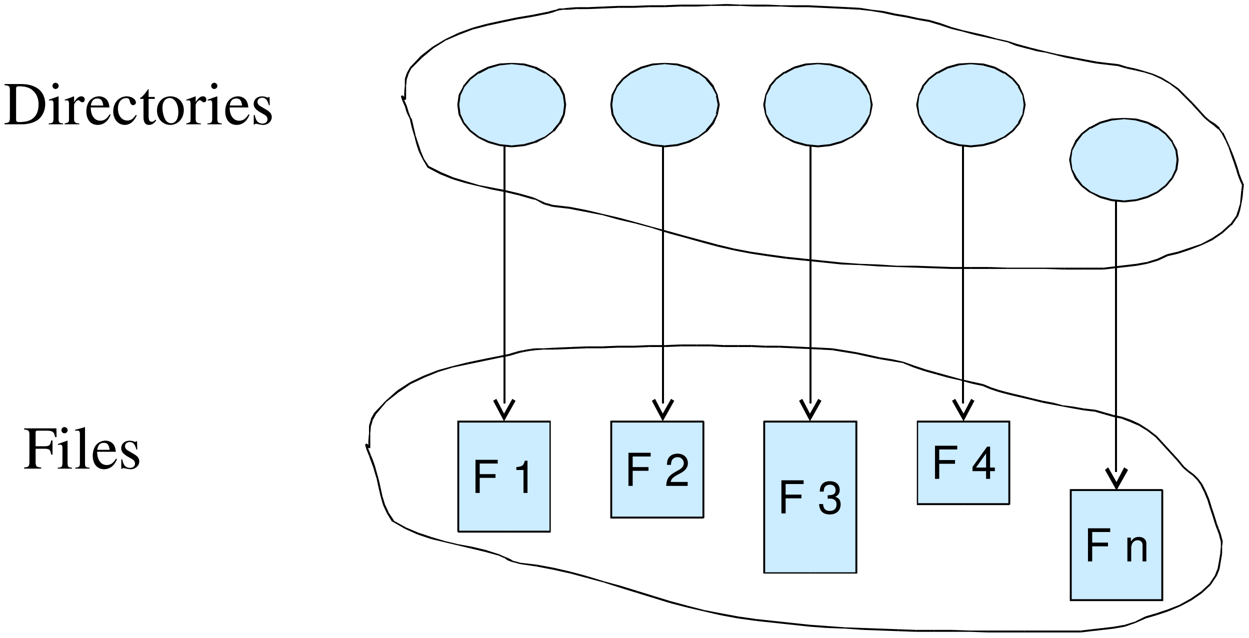
Get attributes: Returns file attributes for processing.

Set attributes: To set the user settable attributes when file changed.

Rename: Rename file.

Directory Structure

A directory is a node containing informations about files.



Directories

Files

Directories can have different structures

Directory Structure

Single-Level-Directory:

All files are contained in the same directory.

Easy to support and understand; but difficult to manage large amount of files and to manage different users.

Two-Level-Directory:

Separate directory for each user.

Used on a multiuser computer and on a simple network computers.

It has problem when users want to cooperate on some task and to access one another's files. It also cause problem when a single user has large number of files.

Directory Structure

Hierarchical-Directory:

Generalization of two-level-structure to a tree of arbitrary height.

This allow the user to create their own subdirectories and to organize their files accordingly.

To allow to share the directory for different user acyclic-graph-structure is used.

Nearly all modern file systems are organized in this manner.

Path Names

Absolute and Relative path names.

Absolute Path Name:

Path name starting from root directory to the file.

e.g. In Unix: /usr/user1/bin/lab2.

Path separated by / in Unix and \ in windows.

Relative Path Name:

Concept of working directory.

A user can designate one directory as the current working directory, in which case all path names not beginning at the root directory are taken relative to the working directory.

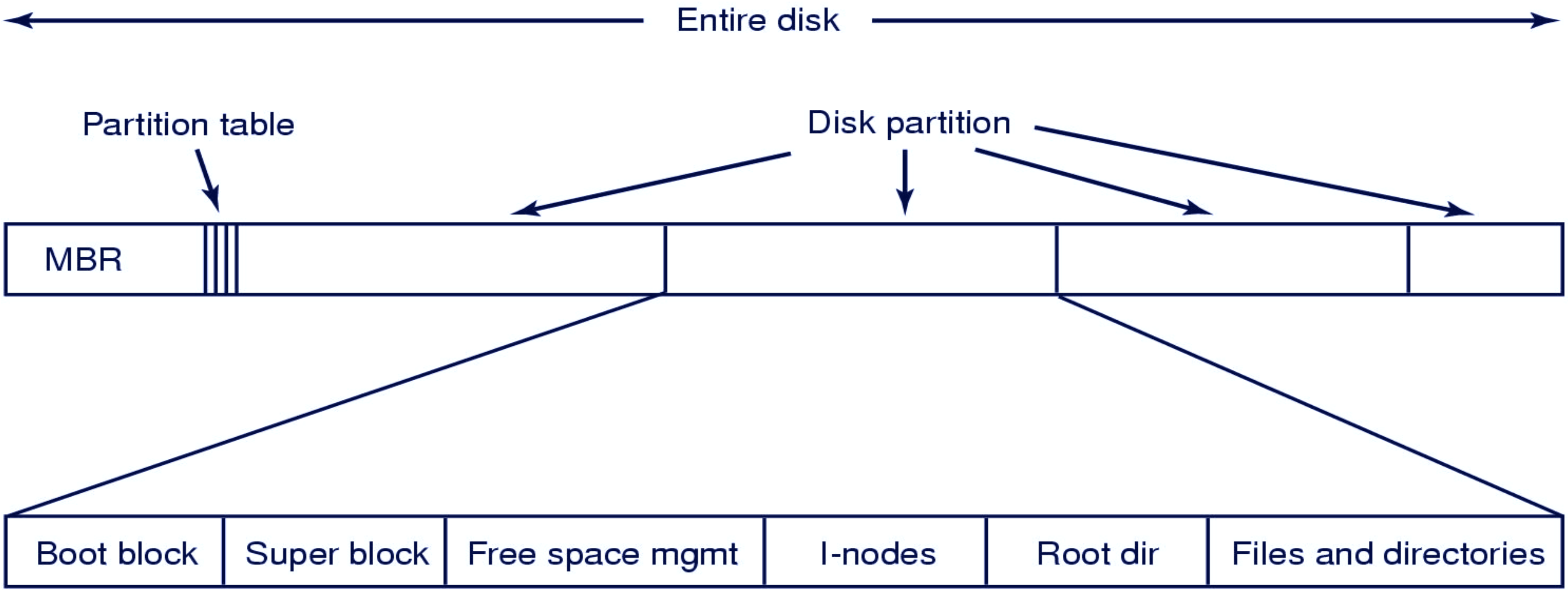
E.g., bin/lab2 is enough to locate same file if current working directory is /usr/user1.

File-System Implementation

How files and directories are stored?
How disk space is managed?
How to make every thing work efficiently and reliably?

File-System Layout

Disks are divided up into one or more partitions, with independent file system on each partition.



Allocation Methods

Contiguous Allocation

Each file occupy a set of contiguous block on the disk.

Disk addresses define a linear ordering on the disk.

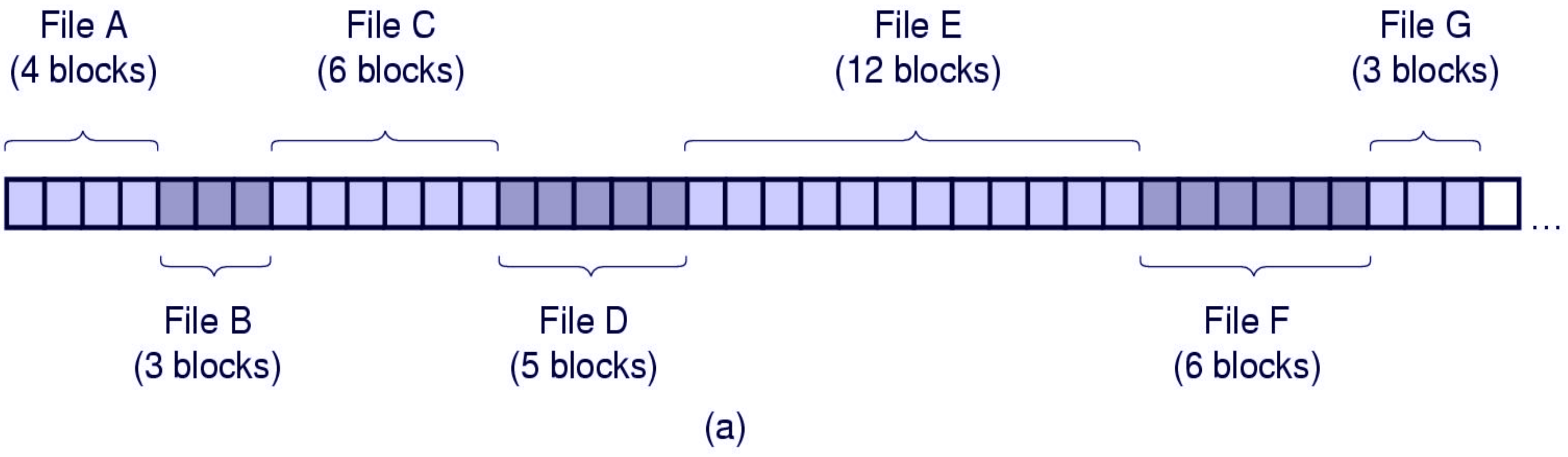
File is defined by the disk address and length in block units.

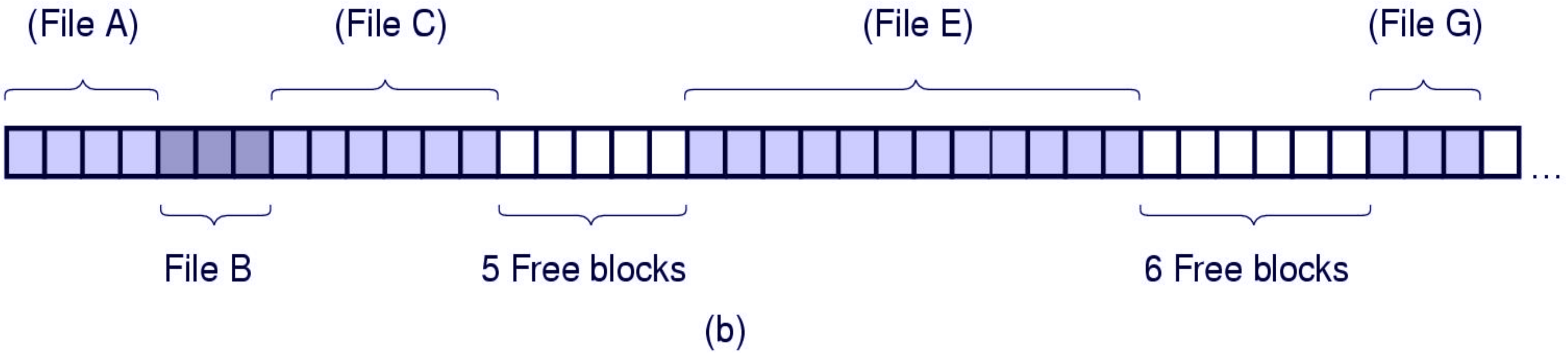
With 2-KB blocks, a 50-KB file would be allocated 25 consecutive blocks.

Both sequential and direct access can be supported by contiguous allocation.

Allocation Methods

Contiguous Allocation





Allocation Methods

Contiguous Allocation

Advantages:

Simple to implement; accessing a file that has been allocated contiguously is easy.

High performance; the entire file can be read in single operation i.e. decrease the seek time.

Problems:

fragmentation: when files are allocated and deleted the free disk space is broken into holes.

Dynamic-storage-allocation problem: searching of right holes.

Required pre-information of file size.

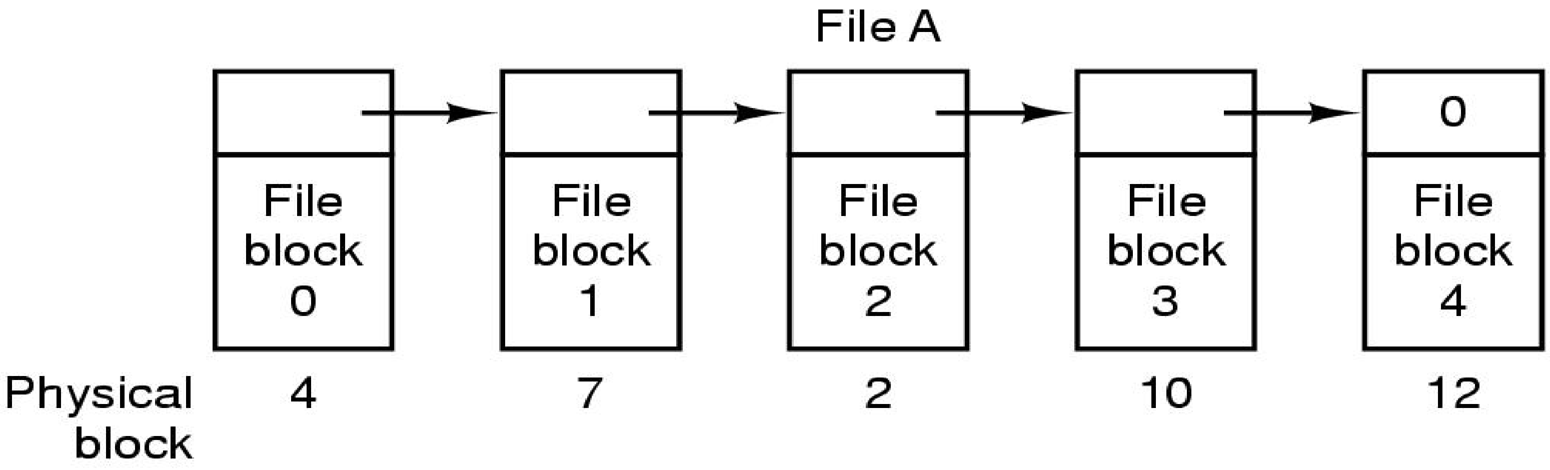
Due to its good performance it used in many system; it is widely used in CD-ROM file system.

Allocation Methods

Linked Allocation

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

Each block contain the pointer to the next block of same file. To create the new file, we simply create a new entry in the directory; with linked allocation, each directory entry has a pointer to the first disk block of the file.



Allocation Methods

Linked Allocation

Problems:

It solves all problems of contiguous allocation but it can used only for sequential access file; random access is extremely slow.

Each block access required disk seek.

It also required space for pointer.

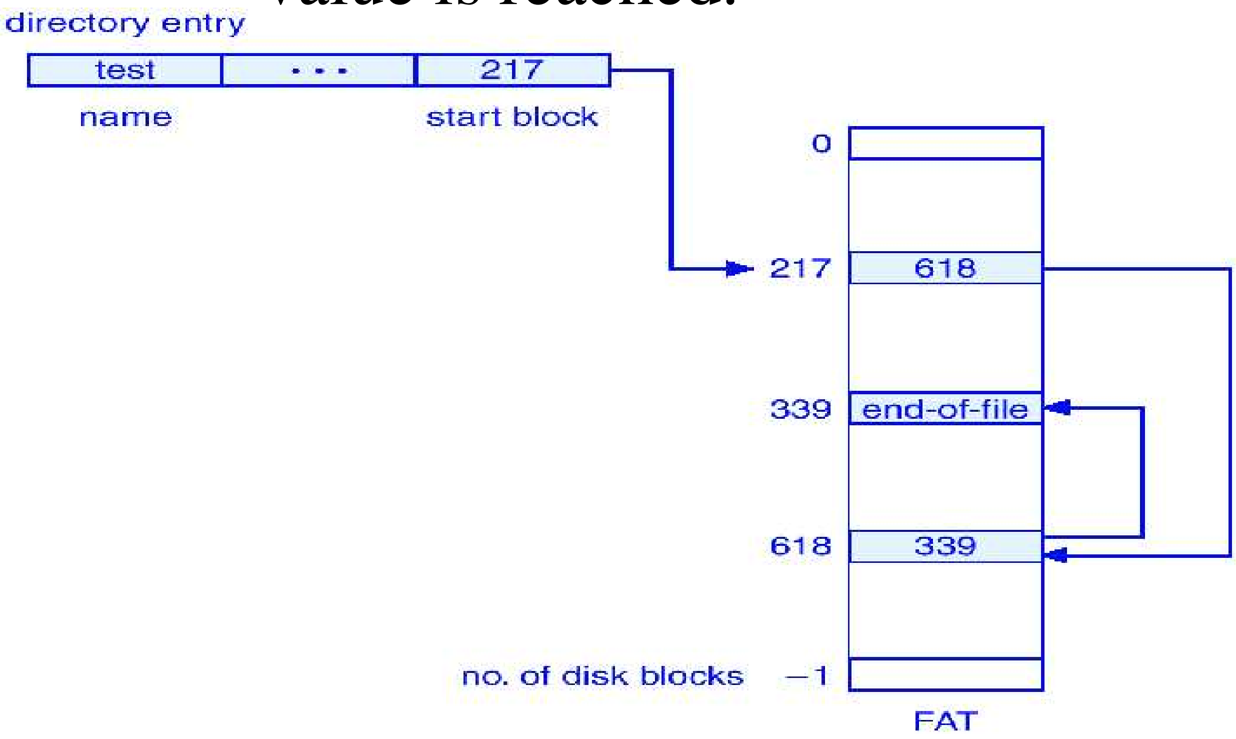
Solution: Using File Allocation Table (FAT).

The table has one entry for each disk block containing next block number for the file. This resides at the beginning of each disk partition.

Allocation Methods

Linked Allocation using FAT

The FAT is used as is a linked list. The directory entry contains the block number of the first block of the file. The FAT is looked to find next block until a special end-of-file value is reached.



Allocation Methods

Linked Allocation using FAT

Advantages:

The entire block is available for data.

Result the significant number of disk seek; random access time is improved.

Problems:

The entire table must be in memory all the time to make it work.

With 20GB disk and 1KB block size, the table needs 20 millions entries. Suppose each entry requires 3 bytes. Thus the table will take up 60MB of main memory all the time.

Allocation Methods

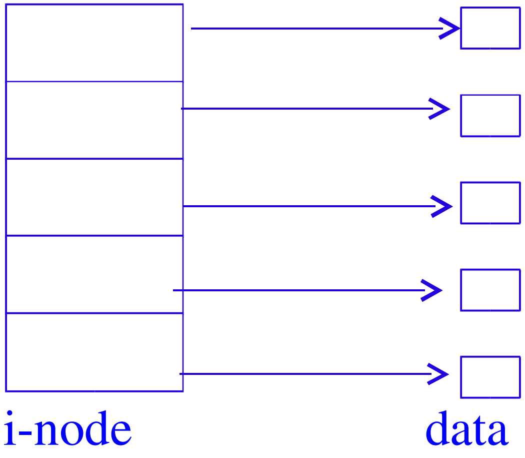
Index Allocation

To keep the track of which blocks belongs to which file, each file has data structure (i-node) that list the attributes and disk address of the disk block associate with the file.

Each i-node are stored in a disk block, if a disk block is not sufficient to hold i-node it can be multileveled.

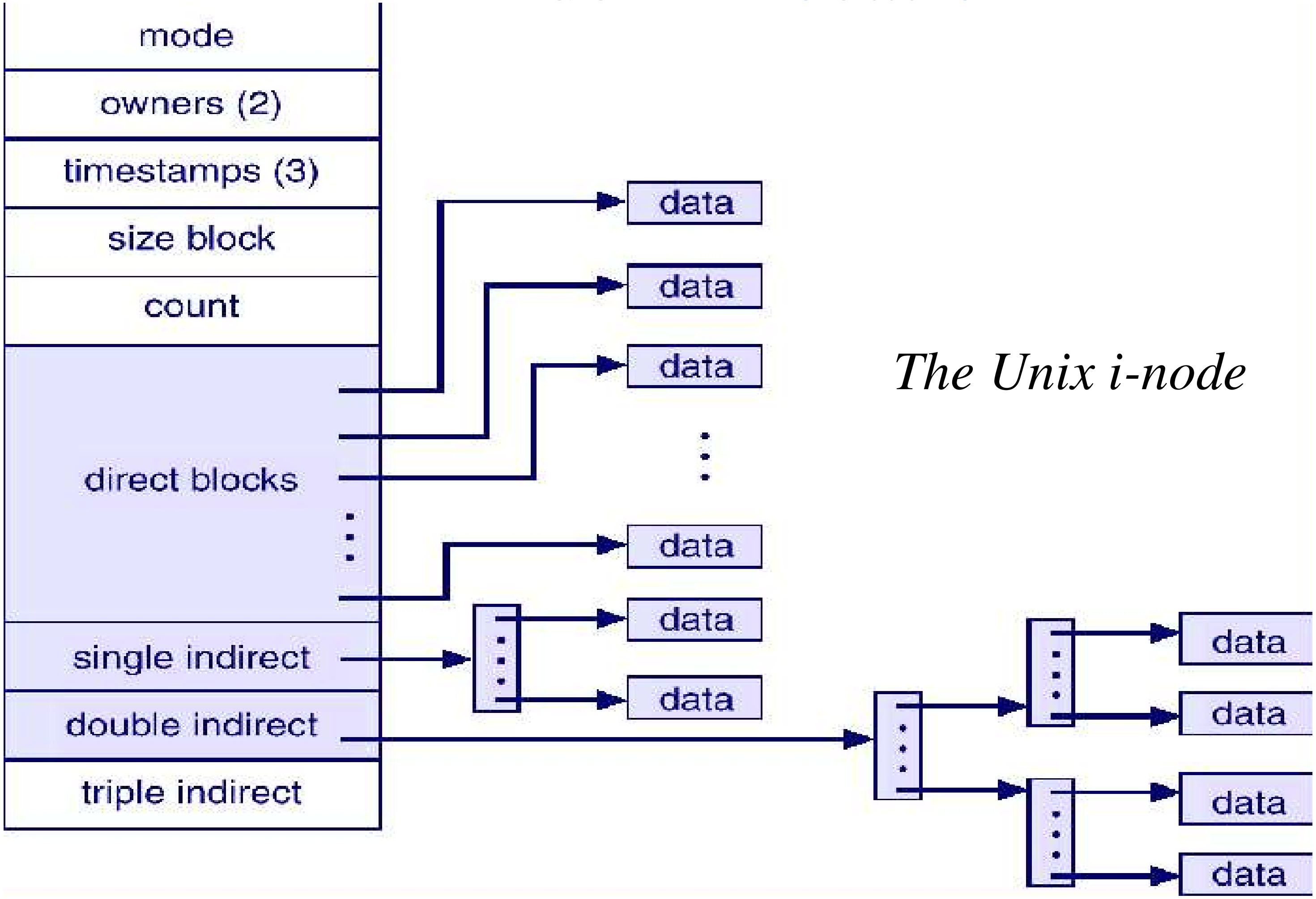
Independent to disk size.

If i-node occupies n bytes for each file and k files are opened, the total memory by i-nodes is kn bytes.



Allocation Methods

Index Allocation



The Unix i-node

Advantages: Far smaller than space occupied by FAT.

Problems: This can be suffer from performance.

Directory Implementation

The directory entry provides the information needed to find the disk block of the file. The file attributes are stored in the directory.

Linear List

Use the linear list of the file names with pointer to the data blocks.

It requires linear search to find a particular entry.

Advantages: Simple to implement.

Problem: linear search to find the file.

Directory Implementation

Hash Table

It consist linear list with hash table.

The hash table takes a value computed from the file name and returns a pointer to the file name in the linear list.

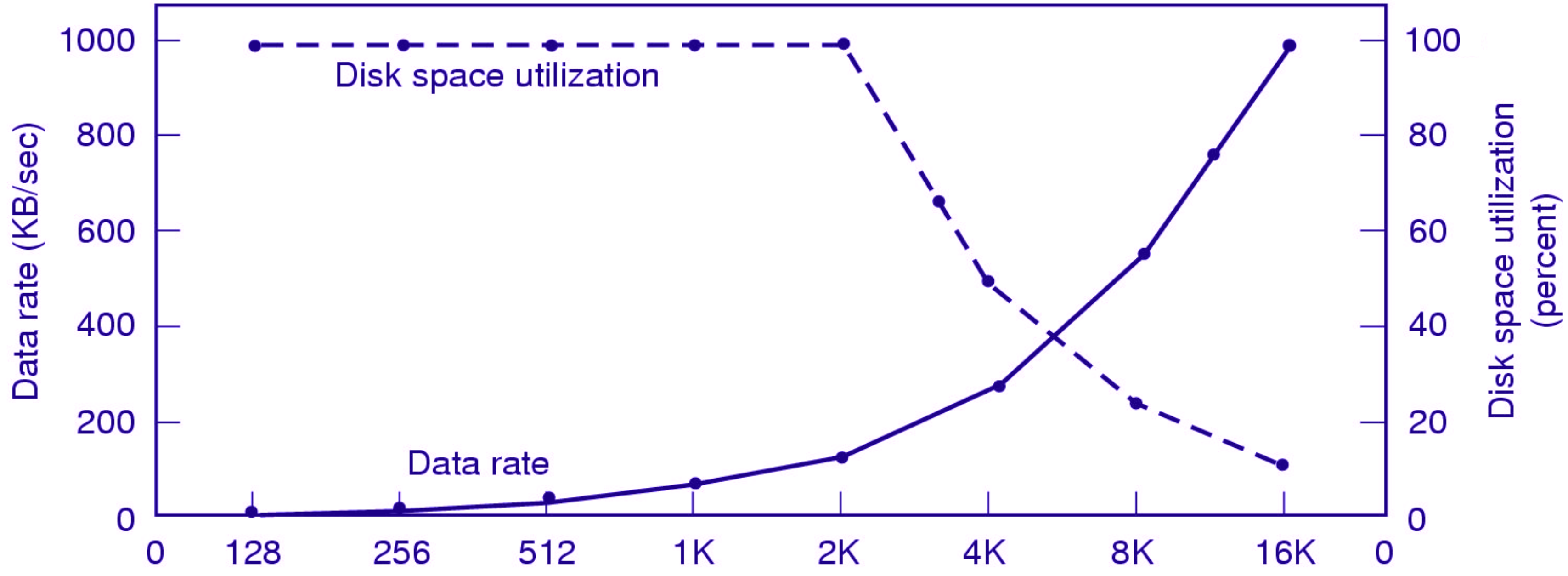
If that key is already in use, a linked list is constructed.

Advantages: greatly decrease the file search time.

Problem: It greatly fixed size and dependence of the hash function on that size.

Block Size

Nearly all file systems chop files up into fixed-size block.
Using a large size result the wastage of disk space.
Using a small size increase the seek and rotational delay- low data access rate.



Dark line (left hand scale) gives data rate of a disk; Dotted line (right hand scale) gives disk space utilization. All files are 2KB

Free-Space Management

To keep track of free blocks, system maintains the free-space-list.

The free-space-list records all free blocks- those not allocated to some file or directory.

To create a file, system search the free-space-list for required amount of space, and allocate that space to the new file and removed from free-space-list.

When a file is deleted, its disk space is added to the free-space-list.

The free-space-list can be implemented in two ways:

Bitmap

Linked list

Free-Space Management

Bitmap

Each block is represented by a bit. If the block is free, the bit is 1; if the block is allocated the bit is 0.

A disk with n blocks requires a bitmap with n bits.

Ex: Consider a disk where blocks 2,3,4,5,8,9,10,11,12,13,17,18,...... are free, and rest are allocated. The free space bit map would be

001111001111110001 0.........

Advantages: Simple and efficient in finding first free block, or n consecutive free blocks.

Problems: Inefficient unless the entire bitmap is kept in main memory.

Keeping in main memory is possible only for small disk, when disk is large the bitmap would be large.

Many system use bitmap method.

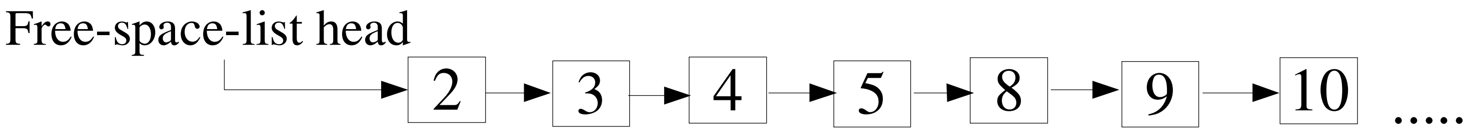
Free-Space Management

Linked List

Link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory.

The first block contains a pointer to the next free block.

The previous example can be represented in linked as



Advantages: Only one block is kept in memory.

Problems: Not efficient; to traverse list, it must read each block.

Home Works

HW #11

1. 6, 7, 10, 15, 20, 22, 36 and 42 from your Textbook (Tanenbaum) Ch. 6.
2. How many bits would be needed to store the free-space-list under the following conditions if a bitmap were used to implement?

   a) 500, 000 blocks total and 200, 000 free blocks.

   b) 1,000,000 blocks total and 0 free blocks.

   Also find how much space is required if it need to be stored in memory?