DISTRIBUTED FILE SYSTEMS

Reference: “Distributed Systems Concepts and Design”, 5th Edition,
G. Coulouris, J. Dollimore and T. Kindberg

BY

S.REVATHY
BE- CSE - III Year

INTRODUCTION

- Sharing of stored information is the most important aspect of distributed resource sharing.
- Some of the problems are load balancing, reliability, availability and security.
- **Distributed File Systems** support sharing of information in the form of files and hardware resources.

INTRO(cntd)

- File systems provide a convenient programming interface to disk storage.
- The most effective in providing shared persistent storage for use in intranets.

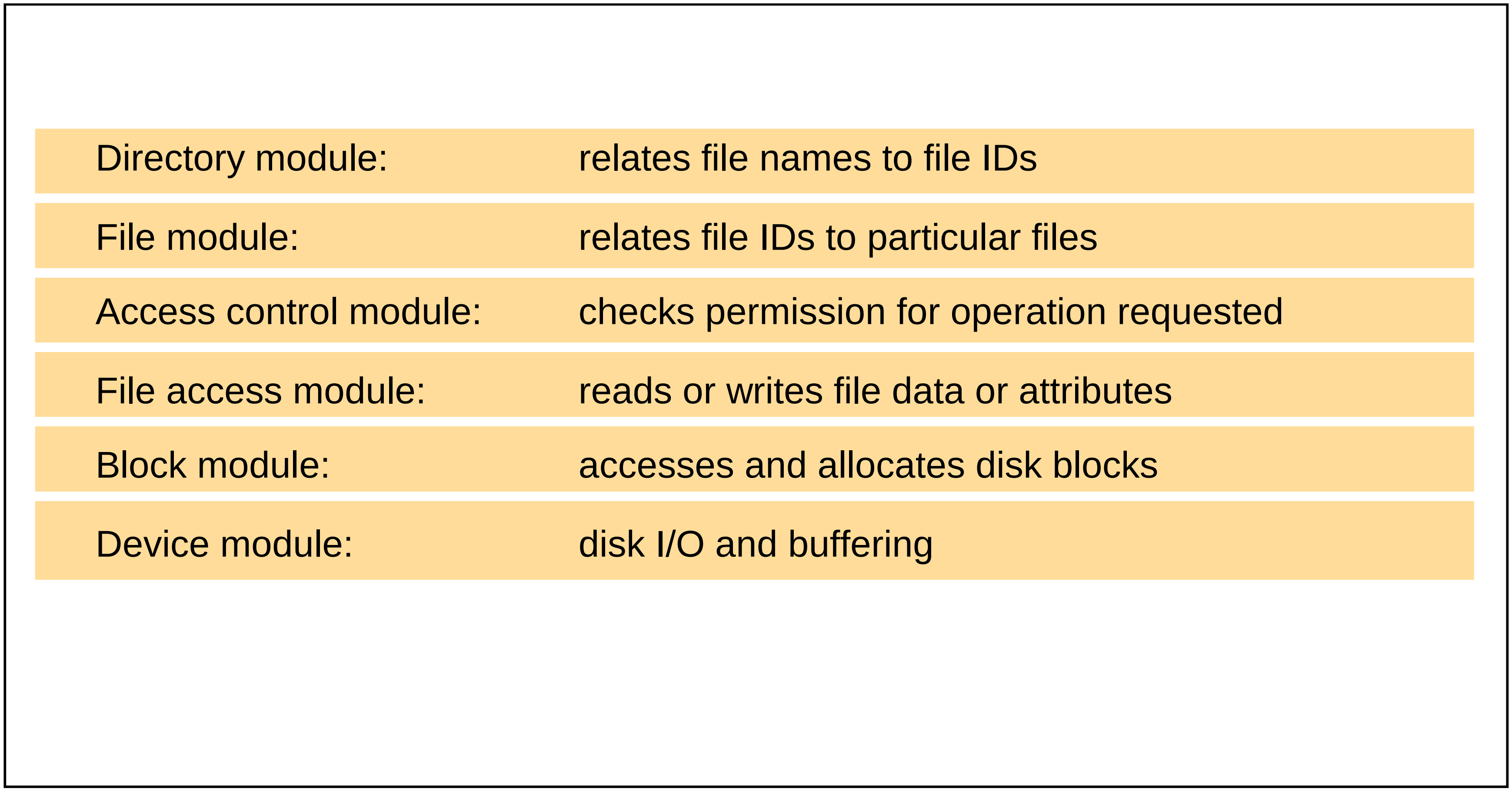
INTRO(cntd)

	Sharing	Persistence	Distributed cache	*Consistency maintenance*	*Example*
Main memory	✗	✗	✗	1	RAM
File system	✗	✓	✗	1	UNIX file system
Distributed file system	✓	✓	✓	✓	Sun NFS
Web	✓	✓	✓	✗	Web server
Distributed shared memory	✓	✗	✓	✓	Ivy (Ch. 18)
Remote objects (RMI/ORB)	✓	✗	✗	1	CORBA
Persistent object store	✓	✓	✗	1	CORBA Persistent Object Service
Peer-to-peer storage system	✓	✓	✓	✓	OceanStore(Ch. 10)

Types of consistency between copies: 1 - strict one-copy consistency
√ - approximate consistency
X - no automatic consistency

Figure 1 Storage systems and their properties

INTRO(cntd)



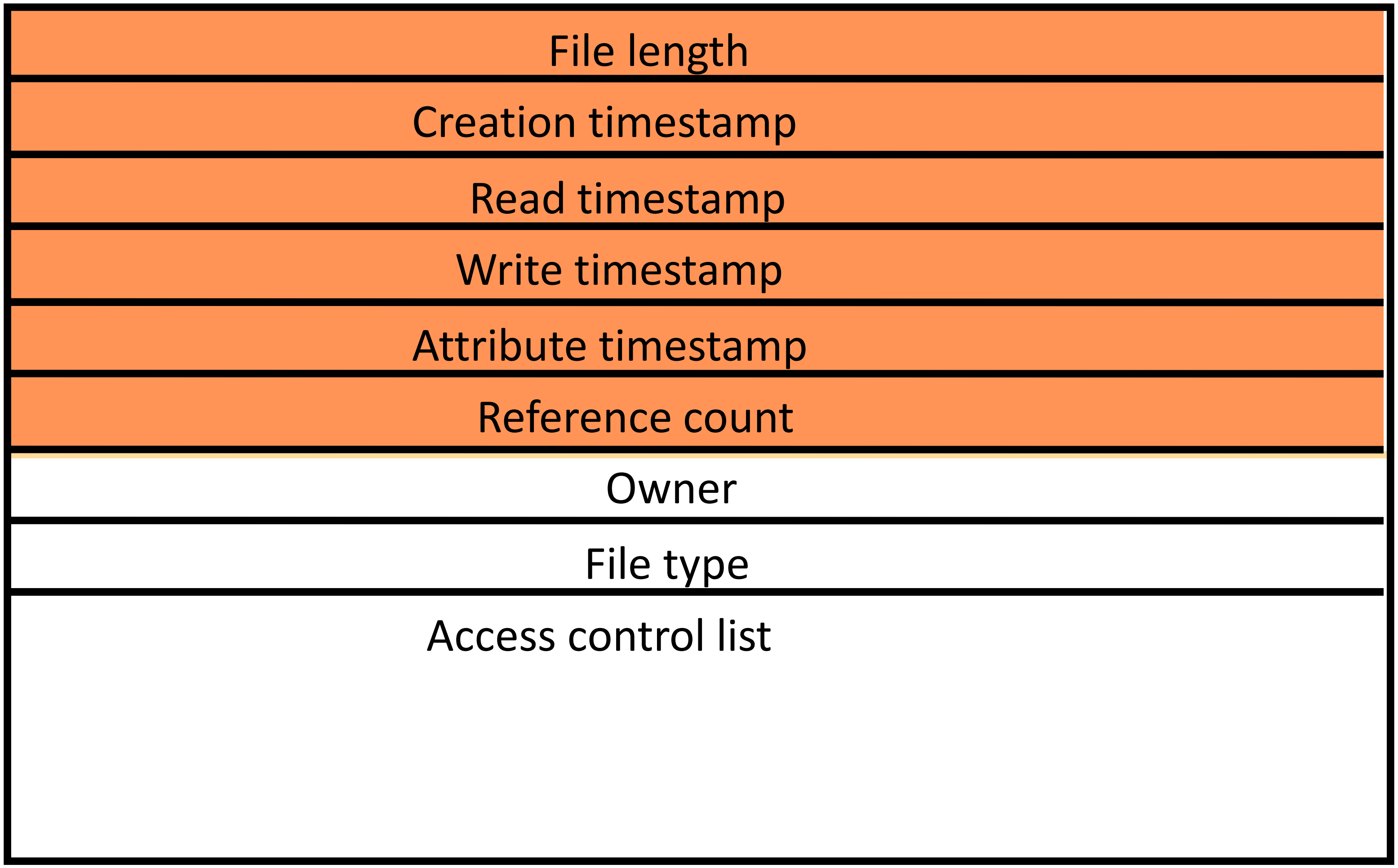
Directory module:	relates file names to file IDs
File module:	relates file IDs to particular files
Access control module:	checks permission for operation requested
File access module:	reads or writes file data or attributes
Block module:	accesses and allocates disk blocks
Device module:	disk I/O and buffering

Figure 2 File System Modules

CHARACTERISTICS OF FILE SYSTEMS

- File systems responsible for the organization, storage, retrieval, naming , sharing and protection of files.
- Programming interface that characterizes the file abstraction.
- Files contain both **data** and **attributes.**
- **Metadata –** extra information stored by a file system for management of files.

File attribute Record Structure



File length

Creation timestamp

Read timestamp

Write timestamp

Attribute timestamp

Reference count

Owner

File type

Access control list

FILE SYSTEM OPERATIONS

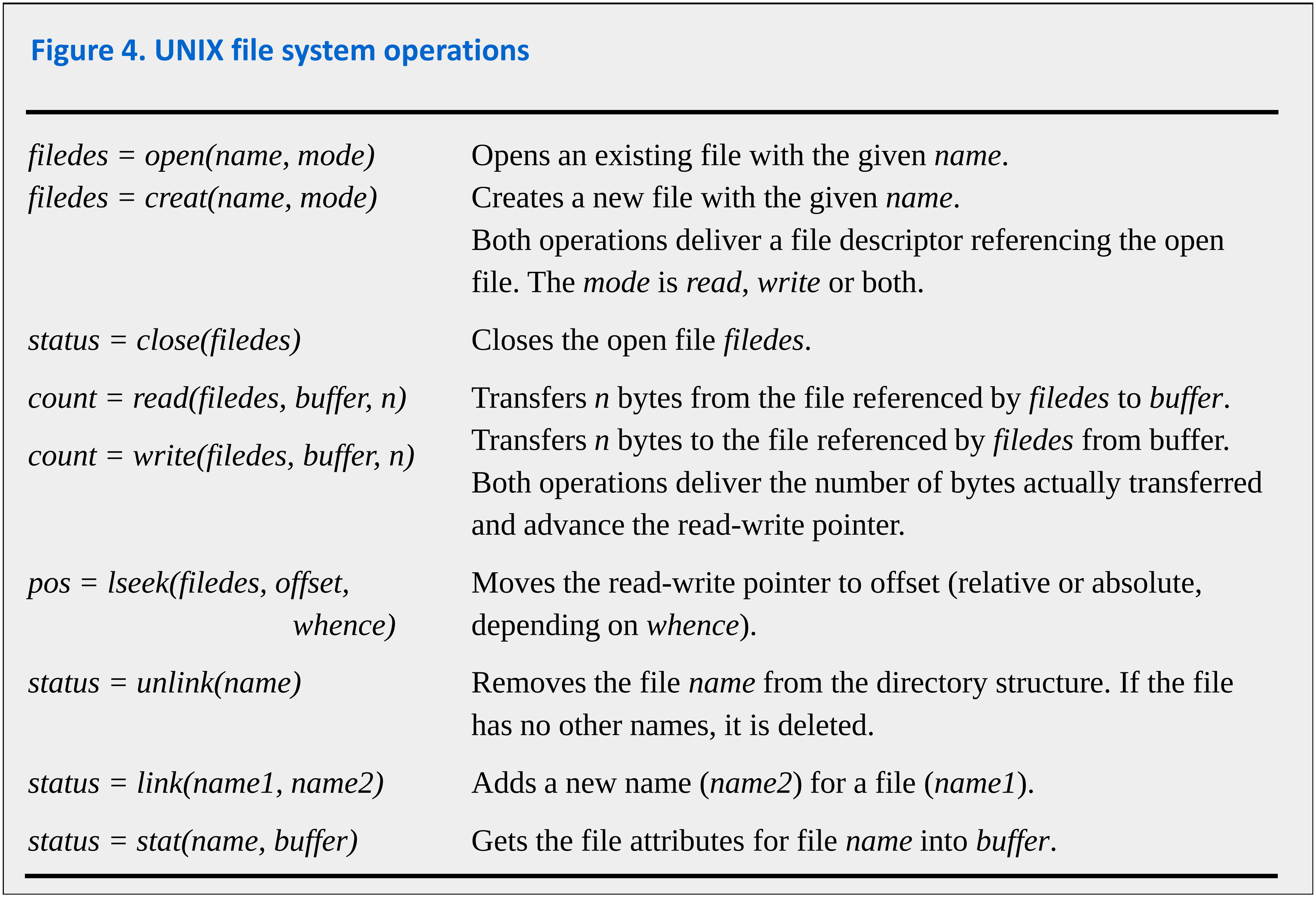


Figure 4. UNIX file system operations

filedes = open(name, mode) *filedes = creat(name, mode)*	Opens an existing file with the given *name*. Creates a new file with the given *name*. Both operations deliver a file descriptor referencing the open file. The *mode* is *read*, *write* or both.
status = close(filedes)	Closes the open file *filedes*.
count = read(filedes, buffer, n) *count = write(filedes, buffer, n)*	Transfers *n* bytes from the file referenced by *filedes* to *buffer*. Transfers *n* bytes to the file referenced by *filedes* from buffer. Both operations deliver the number of bytes actually transferred and advance the read-write pointer.
pos = lseek(filedes, offset, whence)	Moves the read-write pointer to offset (relative or absolute, depending on *whence*).
status = unlink(name)	Removes the file *name* from the directory structure. If the file has no other names, it is deleted.
status = link(name1, name2)	Adds a new name (*name2*) for a file (*name1*).
status = stat(name, buffer)	Gets the file attributes for file *name* into *buffer*.

DISTRIBUTED FILE SYSTEM REQUIREMENTS

- Transparency

  Design of file service should support transparency requirements .

- Concurrent File updates

  Changes to a file should not interfere with operation of other clients accessing the same file .

DISTRIBUTED FILE SYSTEM REQUIREMENTS

- File replication

  A file may be represented by several copies of its contents at different locations

- Hardware and operating system heterogeneity

  Important aspect of openness

- Fault Tolerance

  File service must continue to operate in the face of client and server failures.

- Consistency

  One-copy semantics

DISTRIBUTED FILE SYSTEM REQUIREMENTS

- Security

  Need to authenticate client requests based on user identities and encryption of secret data.

- Efficiency

  Should offer facilities that are at least the same power and generality as found in conventional file systems.

THANK YOU