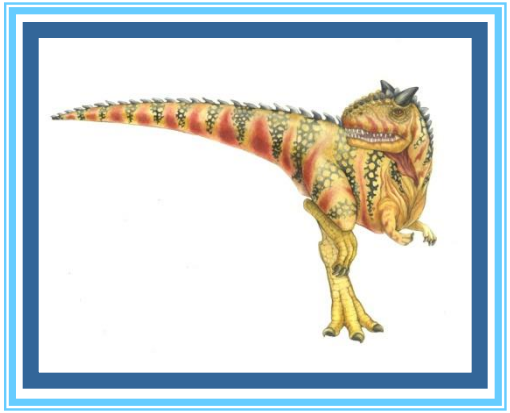
Chapter 10: File-System Interface

الفصل يشرح كيف نظام التشغيل يخزن
الملفات وينظمها ويسمح بالوصول لها
ومشاركتها مع الحفاظ على الحماية.



Silberschatz, Galvin and Gagne ©2009



File Concept

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

File Concept (مفهوم الملف)
Contiguous logical address space
معناها:
الملف يُنظر له منطقيًا كأنه مساحة متصلة من البيانات.
يعني للمستخدم أو البرنامج:
الملف يبدو كأنه سلسلة مرتبة من البيانات بدون فراغات.
مثال:
A B C D E F
حتى لو فعليًا داخل الهاردسك البيانات متفرقة، نظام التشغيل يجعلها تظهر بشكل متصل.

Types of Files (أنواع الملفات)
1) Data Files
ملفات بيانات.
تُستخدم لتخزين المعلومات.
أنواعها:
A) Numeric
بيانات رقمية.
مثال:
12345
98.6
B) Character
بيانات نصية/حروف.
مثال:
Hello
Operating System
C) Binary
بيانات ثنائية (0 و 1).
غالبًا لا تكون مفهومة للبشر مباشرة.
مثال:
الصور
الفيديو
ملفات البرامج التنفيذية

2) Program Files
ملفات البرامج.
تحتوي أوامر وتعليمات يتم تنفيذها.
مثال:
.exe
.class
.out





File Structure

File Structure (بنية الملف)
يعني:
كيف يتم ترتيب البيانات داخل الملف؟

- 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

1) None Structure
لا يوجد هيكل محدد.
الملف مجرد:
bytes
words
مثل:
A B C D E
نظام التشغيل لا يهتم بالمعنى.
البرنامج نفسه يفسر البيانات.

2) Simple Record Structure
هيكل بسيط للملف.
البيانات تكون على شكل Records.
A) Lines
البيانات مرتبة كسطور.
مثال:
Ali
Sara
Omar
مثل ملفات txt.
B) Fixed Length
كل Record له نفس الطول.
مثال:
ID Name
101 Ali
102 Sara
كل سطر نفس الحجم.
الميزة:
الوصول أسرع وأسهل.
العيب:
قد يهدر مساحة.
C) Variable Length
كل Record حجمه مختلف.
مثال:
Ali
Mohammed Al-Qahtani
Sara
الميزة:
يوفر مساحة.
العيب:
إدارته أصعب.

3) Complex Structures هياكل معقدة
A) Formatted Document
ملفات تحتوي تنسيق.
مثل:
Word
PDF
تشمل:
خطوط
الوان
صور
B) Relocatable Load File
ملفات تنفيذية يمكن تحميلها في أماكن مختلفة بالذاكرة.
مرتبطة بالـ Loader والـ Linking.

يعني حتى الملفات المعقدة يمكن تمثيلها كسلسلة Bytes عادية باستخدام:
control characters
special markers

مين يحدد شكل الملف؟
1) Operating System
أحيانًا النظام يفرض هيكل معين.
2) Program
أحيانًا البرنامج نفسه يحدد.
مثال:
Word يحدد شكل .docx
Photoshop يحدد شكل .psd





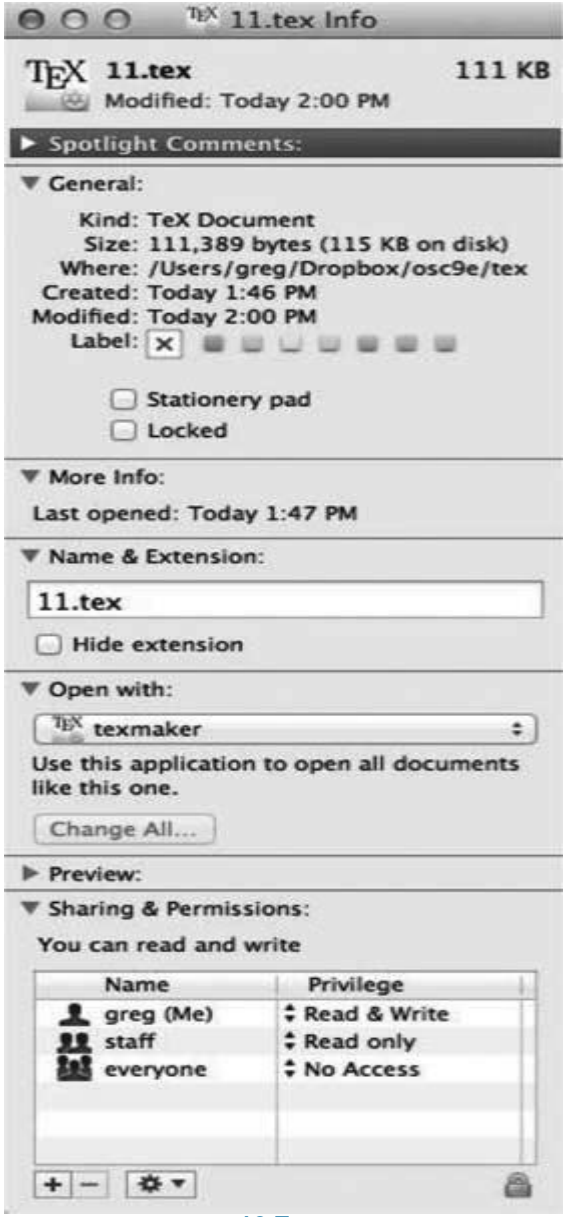
File Attributes

- **Name** – only information kept in human-readable form
  اسم الملف وهو الشيء الوحيد المقروء للبشر.
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
  نوع الملف.
  مثل: .txt
  يساعد النظام يعرف كيف يتعامل معه.
- **Location** – pointer to file location on device
  مكان الملف في التخزين.
  يعني وين موجود على الهاردسك.
- **Size** – current file size
  حجم الملف الحالي.
- **Protection** – controls who can do reading, writing, executing
  صلاحيات الملف.
- **Time, date, and user identification** – data for protection, security, and usage monitoring
  معلومات إضافية مثل:
  تاريخ الإنشاء
  آخر تعديل
  صاحب الملف
  تستخدم في:
  الحماية
  الأمان
  المراقبة
- Information about files are kept in the directory structure, which is maintained on the disk
- Many variations, including extended file attributes such as file checksum
- Information kept in the directory structure

  Directory Structure
  كل هذه المعلومات تُخزن داخل:
  Directory Structure
  يعني المجلدات تحتفظ بمعلومات الملفات.
  Extended File Attributes
  بعض الأنظمة تضيف خصائص إضافية.
  مثل:
  File Checksum
  قيمة تستخدم للتأكد أن الملف لم يتغير أو يتلف.



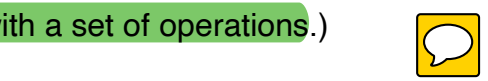
File info Window on Mac OS X





File Operations

- File is an **abstract data type** (ADT is a set of objects together with a set of operations.)
- **Create**
- **Write**
- **Read**
- **Reposition within file**
- **Delete**
- **Truncate (cut off)** تفريغ محتوى الملف بدون حذف الملف نفسه.
- *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



الملف يُعتبر:
Abstract Data Type (ADT)
يعني:
لدينا "كائن" اسمه File
وله مجموعة عمليات نستطيع تنفيذها عليه
مثل فكرة الـ class في البرمجة الكائنية.
مثال بسيط
الملف ليس فقط بيانات، بل أيضًا عمليات مثل:
create
read
write
delete

ماذا يحدث عند Open؟
النظام:
يبحث عن الملف داخل الـ directory على القرص، يجلب
معلوماته للذاكرة، ثم يضعه ضمن الملفات المفتوحة
يعني معلومات الملف تنتقل من:
Disk
إلى:
Main Memory
لتسريع الوصول.

ماذا يحدث عند Close؟
النظام:
يحفظ أي تعديلات
ينقل معلومات الملف من الذاكرة إلى القرص
يزيله من قائمة الملفات المفتوحة





Open Files

- Several pieces of data are needed to manage open files:
  - File pointer: pointer to last read/write location, per process that has the file open
  - File-open count: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it
  - Disk location of the file: cache of data access information
  - Access rights: per-process access mode information

عندما يفتح Process ملفًا، نظام التشغيل يحتاج يحتفظ بمعلومات عنه داخل الذاكرة.

هذه المعلومات تُخزن غالبًا في:

Open File Table

ما البيانات التي يحتاجها النظام؟

1) File Pointer

مؤشر الملف.

يشير إلى:

آخر مكان تمت قراءته أو الكتابة فيه.

مثال

لدينا ملف:

A B C D E

إذا قرأنا:

A

فالمؤشر ينتقل إلى:

B

حتى يعرف النظام أين يكمل القراءة لاحقًا.

مهم:

كل Process لها File Pointer خاص بها.

يعني برنامجين يفتحون نفس الملف:

كل واحد يقرأ بشكل مستقل.

2) File-open count

عداد عدد مرات فتح الملف.

لماذا نحتاجه؟

إذا أكثر من Process فتحوا نفس الملف:

النظام يحتاج يعرف:

كم عملية ما زالت تستخدم الملف؟

مثال

Process A opens file

Process B opens file

العداد يصبح:

2

إذا A أغلق الملف:

count = 1

النظام لا يحذف معلومات الملف من الذاكرة إلا عندما:

count = 0

3) Disk Location of the File

مكان الملف على القرص.

النظام يحتفظ بمعلومات موقع الملف لتسريع الوصول.

بدل البحث كل مرة داخل الهاردسك.

4) Access Rights

صلاحيات الوصول.

لكل Process صلاحيات معينة مثل:

Read

Write

Execute





Open File Locking

- Provided by some operating systems and file systems
- Mediates (manage) access to a file
- Mandatory (compulsory) or advisory (consultative):
  - **Mandatory** – access is denied depending on locks held and requested
  - **Advisory** – processes can find status of locks and decide what to do





File Types – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information





Access Methods

كيف نقرأ أو نكتب البيانات داخل الملف؟

- **Sequential Access**

  read next
  write next
  reset
  no read after last write
  (rewrite)

- **Direct Access**

  read *n*
  write *n*
  position to *n*
  read next
  write next
  rewrite *n*

  n = relative block number

1) Sequential Access
الوصول التسلسلي.
البيانات تُقرأ بالترتيب.
العمليات:
read next
اقرأ العنصر التالي.
write next
اكتب في المكان التالي.
reset
ارجع للبداية.
no read after last write
بعد الكتابة لا نستطيع القراءة مباشرة إلا بعد إعادة ضبط المؤشر.
rewrite
إعادة كتابة البيانات.

2) Direct Access
الوصول المباشر.
يمكن الذهاب مباشرة لأي Block.
العمليات
read n
اقرأ البلوك رقم n.
write n
اكتب في البلوك رقم n.
position to n
انقل المؤشر للبلوك n.
rewrite n
أعد كتابة البلوك n.





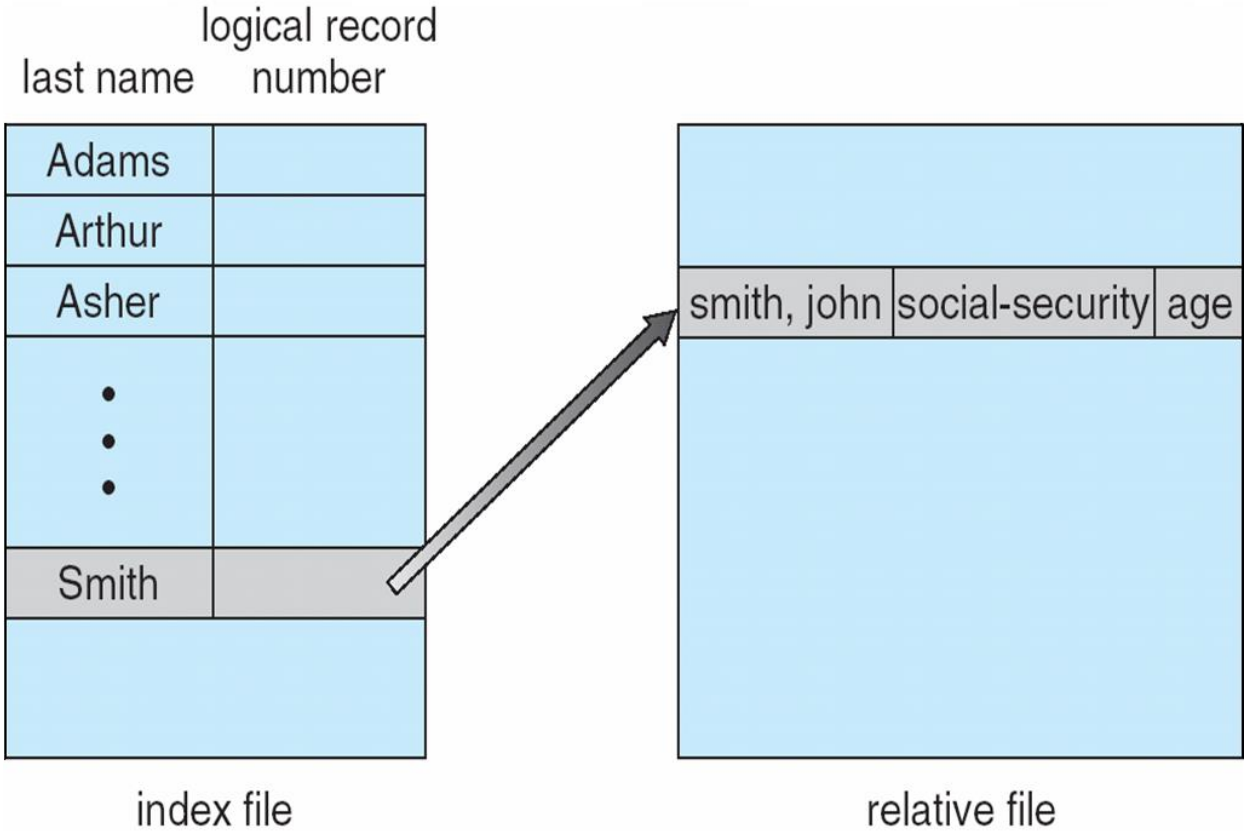
Simulation of Sequential Access on Direct-access File

sequential access	implementation for direct access
reset	$cp = 0;$
read next	*read cp*; $cp = cp + 1;$
write next	*write cp*; $cp = cp + 1;$





Example of Index and Relative Files (P.385)





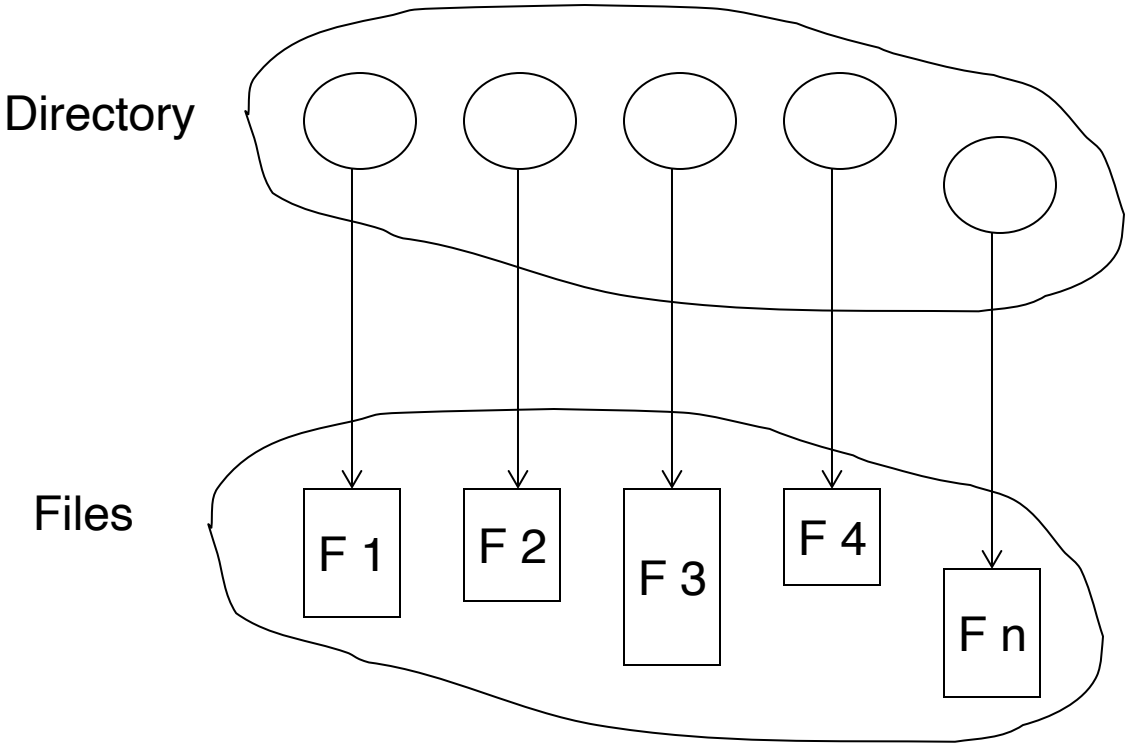


Directory Structure

كيف نظام التشغيل ينظم الملفات داخل الأقراص.

- A collection of nodes containing information about all files

الـ Directory = Folder
وهو:
مجموعة تحتوي معلومات عن الملفات.



الملفات مخزنة على Disk
وكذلك الـ Directories مخزنة على Disk

Both the directory structure and the files reside on disk
Backups of these two structures are kept on tapes





Disk Structure

هذا الجزء يشرح كيف يتم تنظيم الهاردسك نفسه.

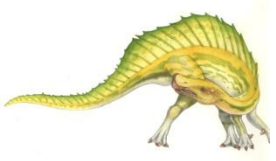
- Disk can be subdivided into partitions
  - الهاردسك يمكن تقسيمه إلى أجزاء اسمها: Partitions
- Disks or partitions can be RAID protected against failure
  - تقنية تستخدم لحماية البيانات من تلف الأقراص.
- Disk or partition can be used raw – without a file system, or formatted with a file system
  - قرص بدون File System النظام لا يستطيع تنظيم الملفات عليه بسهولة.
- Partitions also known as minidisks, slices
  - هناك أسماء أخرى لـ Partitions
- Entity containing file system known as a volume
  - Partition + File System = Volume
- Each volume containing file system also tracks that file system's info in device directory or volume table of contents
- As well as general-purpose file systems there are many special-purpose file systems, frequently all within the same operating system or computer

لماذا نستخدم Partitions؟
تنظيم البيانات
فصل النظام عن الملفات
تحسين الإدارة

قرص يحتوي:
File System
Directories
تنظيم ملفات

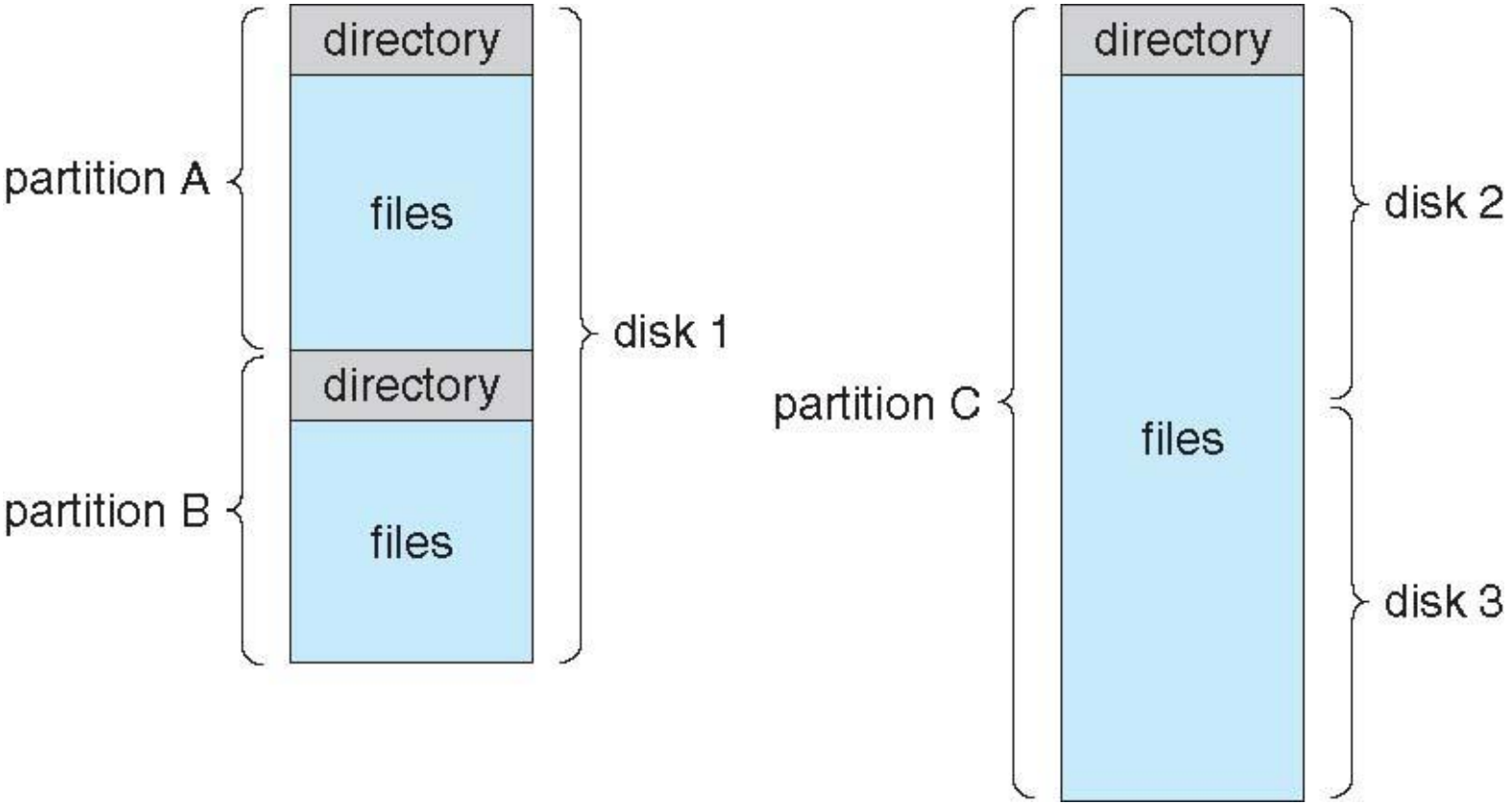
6) Volume Table / Device Directory
كل Volume يحتفظ بمعلومات عن:
الملفات
المساحة
البلوكات
حالة النظام

7) Special-Purpose File Systems
ليس كل الـ File Systems عامة.
بعضها مصمم لغرض خاص.





A Typical File-system Organization







Operations Performed on Directory

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system (tree search)

ما العمليات التي يمكن تنفيذها على الـ Directory؟

1) Search for a File
البحث عن ملف.
مثال:
search report.pdf
2) Create a File
إنشاء ملف جديد داخل المجلد.
3) Delete a File
حذف ملف من المجلد.
4) List a Directory
عرض الملفات داخل المجلد.
مثل أمر:
ls
أو:
dir
5) Rename a File
تغيير اسم الملف.
مثال:
notes.txt → os_notes.txt
6) Traverse the File System
وتسمى:
Tree Search
معناها:
التنقل داخل هيكل الملفات والمجلدات.
مثل:

```
Home
├── Documents
│   └── OS
│       └── chapter10.pdf
```

النظام يتنقل بين المجلدات للوصول للملف.





Organize the Directory (Logically) to Obtain

لماذا ننظم الـ Directory؟

- Efficiency – locating a file quickly
- Naming – convenient to users
  - Two users can have same name for different files
  - The same file can have several different names
- Grouping – logical grouping of files by properties, (e.g., all Java programs, all games, …)

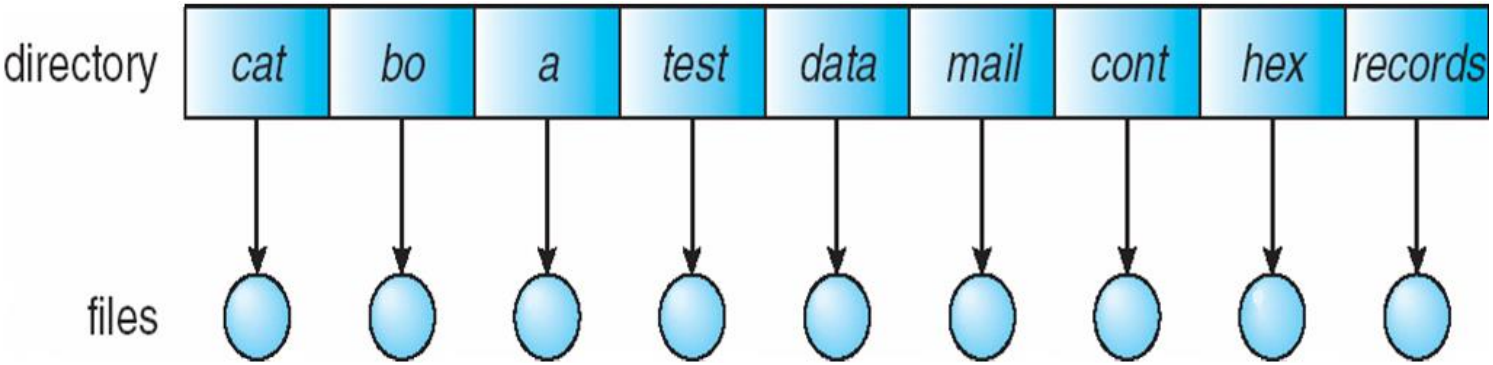
1) Efficiency (الكفاءة)
الهدف:
العثور على الملفات بسرعة.
كلما كان التنظيم أفضل:
البحث أسرع
الوصول أسهل
2) Naming (التسمية)
التنظيم يساعد على:
استخدام أسماء مريحة للمستخدم
تجنب تعارض الأسماء
مثال
شخصان قد يملكان ملفين بنفس الاسم:
Ali/report.txt
Sara/report.txt
هذا طبيعي في الأنظمة الحديثة.
نفس الملف بعدة أسماء
النظام قد يسمح:
لنفس الملف أن يمتلك أكثر من اسم.
وهذا يسمى غالبًا:
Links
3) Grouping (التجميع)
ترتيب الملفات حسب النوع أو الغرض.
أمثلة
Java Programs
Games
Photos
Documents
هذا يسهل:
التنظيم
البحث





Single-Level Directory

- A single directory for all users



Naming problem

Grouping problem

المشاكل

1) Naming Problem

لا يمكن وجود ملفين بنفس الاسم.

مثال

إذا Ali أنشأ:

report.txt

ف Sara لا تستطيع إنشاء:

report.txt

2) Grouping Problem

لا يوجد تنظيم جيد.

كل الملفات مختلطة.

مثال

games

photos

programs

music

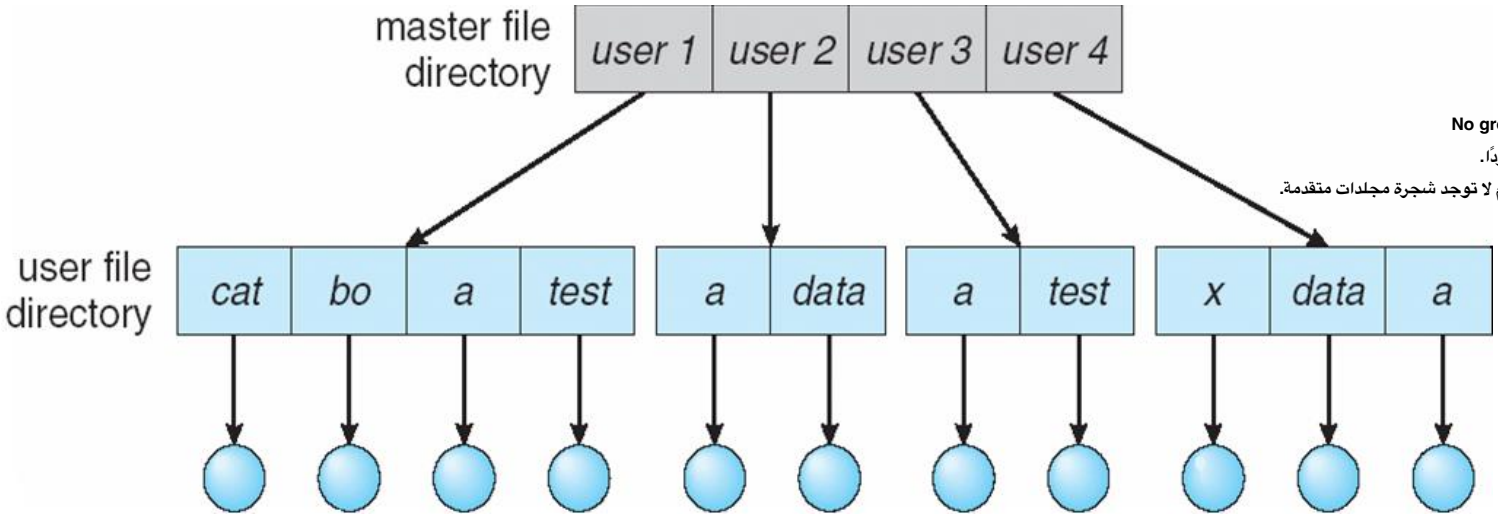
كلها في نفس المكان.





Two-Level Directory

- Separate directory for each user



- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability



المميزات

1) يمكن تكرار نفس الاسم
يمكنهما امتلاك Sara و Ali:
report.txt
بدون مشكلة.

2) Efficient Searching
البحث أسرع لأن كل مستخدم يبحث داخل مجلده فقط.

Path Name
للوصول للملف نستخدم:
Path Name
مثال
/Ali/report.txt
المسار يحدد:
المستخدم
المجلد
الملف

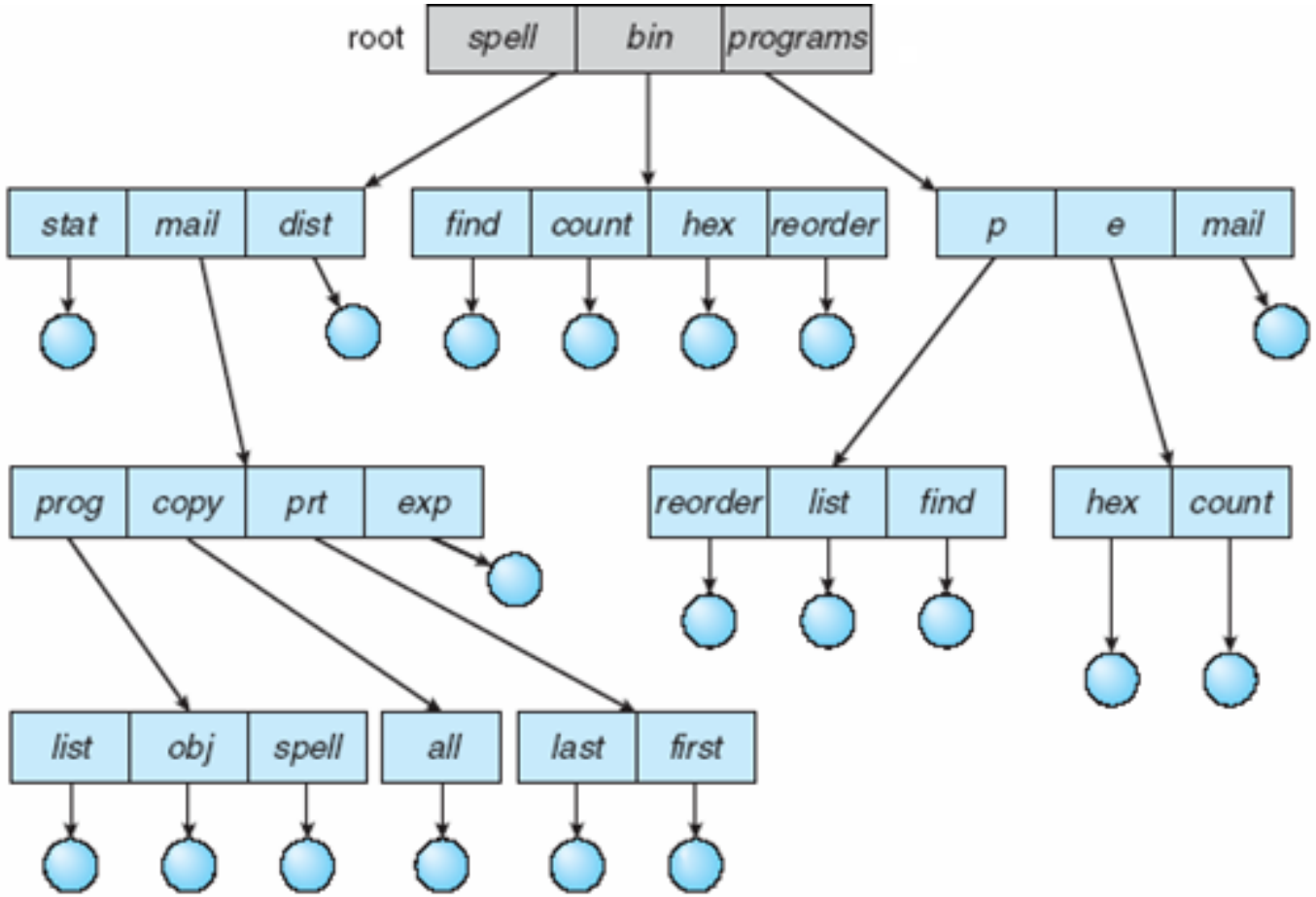
العيب

No grouping capability
لا يزال التنظيم محدودًا.
داخل مجلد المستخدم لا توجد شجرة مجلدات متقدمة.

كل مستخدم يمتلك Directory خاص به.



Tree-Structured Directories







Tree-Structured Directories (Cont.)

- Efficient searching
- Grouping Capability
- Current directory (working directory)
  - cd /spell/mail/prog
  - type list

Tree-Structured Directory

هذا النظام المستخدم حالياً في أغلب الأنظمة.

الفكرة

المجلدات تكون على شكل:

شجرة

مثال

```
Home
├── Documents
│   ├── OS
│   └── Java
├── Games
└── Photos
```

المميزات

✅ تنظيم ممتاز

✅ سهولة البحث

✅ يدعم التجميع

✅ يدعم عدداً كبيراً من الملفات





Tree-Structured Directories (Cont)

- **Absolute** or **relative** path name
- Creating a new file is done in current directory (أي ملف جديد يُنشأ داخل المجلد الحالي)
- Delete a file

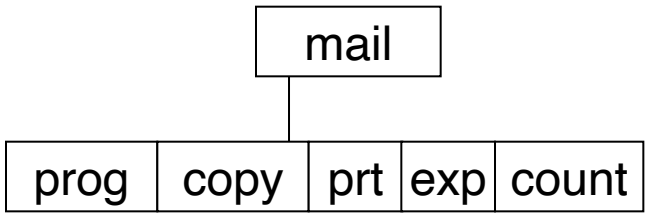
  rm <file-name>
- Creating a new subdirectory is done in current directory

  mkdir <dir-name>

  Example: if in current directory /mail

  mkdir count

1) Absolute Path
المسار الكامل من الـ Root.
مثال
/home/ali/Documents/os.pdf
يبدأ دائمًا من الجذر /
2) Relative Path
المسار بالنسبة للمكان الحالي.
مثال
إذا كنت داخل:
/home/ali
يمكن كتابة:
Documents/os.pdf
بدل المسار الكامل.



Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”





File System Mounting

- A file system must be **mounted** before it can be accessed
- A unmounted file system (i.e., Fig. 10-11(b)) is mounted at a **mount point**

ما معنى Mounting؟

الـ Mounting يعني:

ربط نظام ملفات (File System) بنظام التشغيل حتى يصبح قابلاً للاستخدام.



الفكرة ببساطة

عندما توصل:

USB

CD

Hard Disk

النظام لا يستطيع استخدام الملفات مباشرة.

أولًا يجب أن يقوم بعملية:

Mount

مثال واقعي

عند إدخال USB:

النظام يكتشف الجهاز

يعمل Mount

تظهر الملفات للمستخدم

بدون Mount

النظام يرى:

الجهاز موجود

لكن:

لا يستطيع قراءة الملفات داخله

ما هو Mount Point؟

هو:

المكان الذي يظهر فيه الـ File System داخل شجرة الملفات.

مثال في Linux

لدينا:

/home

/media

/mnt

عند توصيل USB قد يتم تركيبه داخل:

/media/usb

هنا:

/media/usb

يسمى:

Mount Point

لماذا نحتاج Mounting؟

حتى يعرف نظام التشغيل:

أين يبدأ الـ File System

كيف يقرأ الملفات

أين يظهر للمستخدم

فكرة مهمة

Mount Point ≠ Disk

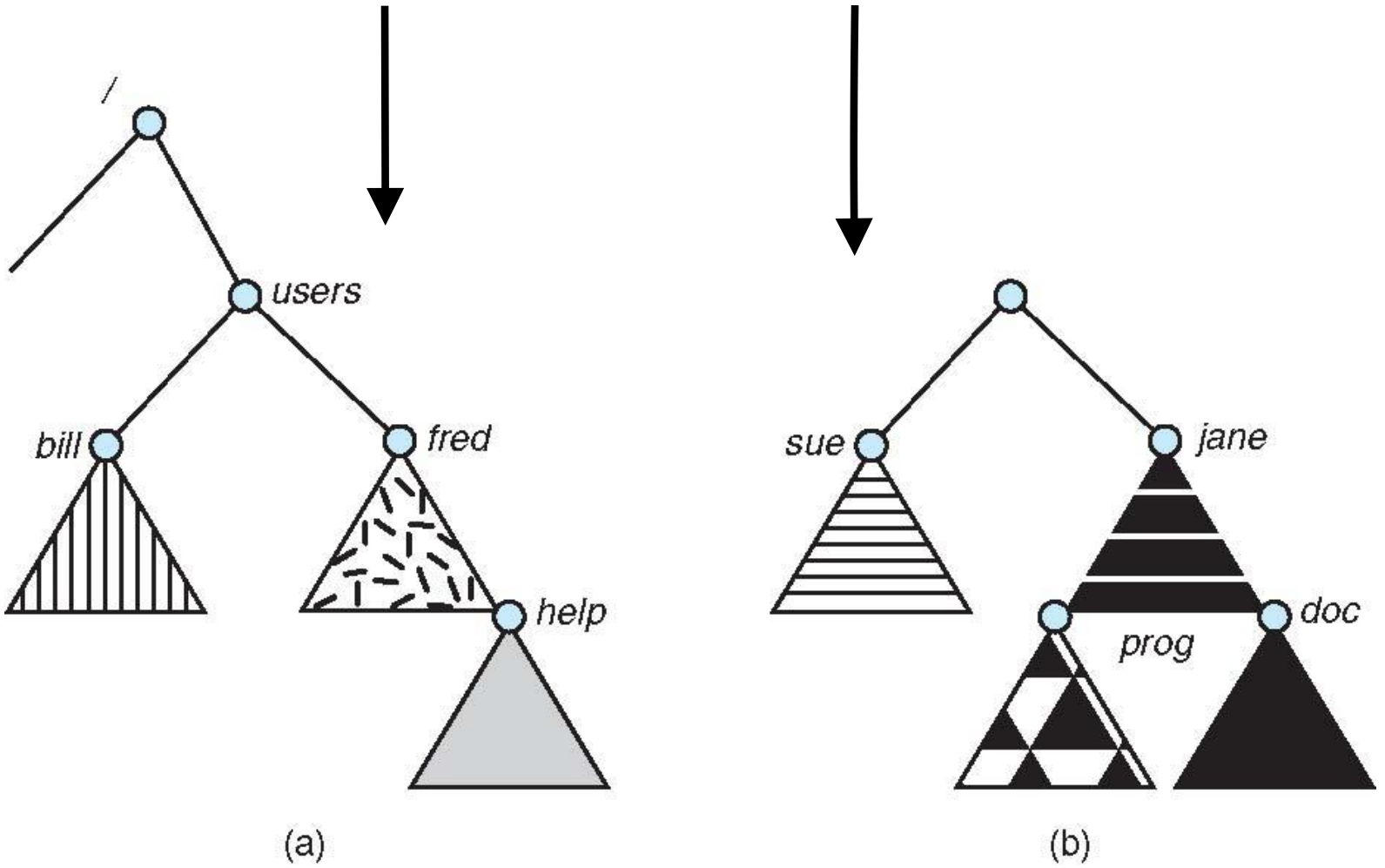
الـ Mount Point مجرد:

مجلد يمثل مكان ظهور الـ File System.





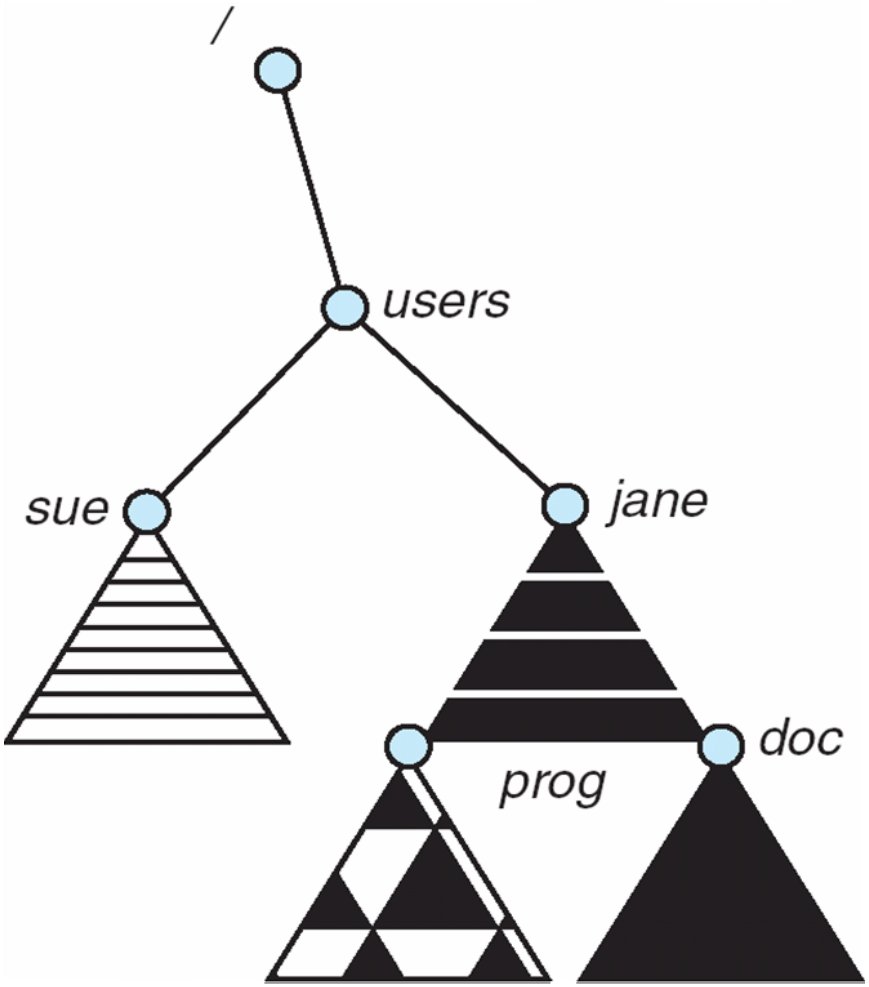
(a) Existing (b) Unmounted Partition







Mount Point







File Sharing

كيف يشارك عدة مستخدمين نفس الملفات مع الحفاظ على الحماية والصلاحيات.

مشاركة الملفات في الأنظمة متعددة المستخدمين شيء مهم ومطلوب.

- Sharing of files on multi-user systems is desirable

تتم المشاركة باستخدام نظام حماية وصلاحيات.

- Sharing may be done through a **protection** scheme

الملفات قد تُشارك بين أجهزة مختلفة عبر الشبكة.

- On distributed systems, files may be shared across a network

- Network File System (NFS) is a common distributed file-sharing method

ما هو NFS؟

وهو نظام يسمح:

بمشاركة الملفات عبر الشبكة وكأنها موجودة محليًا.

الفكرة

الملف موجود على:

Server

لكن المستخدم على جهاز آخر يستطيع فتحه كأنه داخل جهازه.

مثال منطقي

```
Server
└── shared
    └── report.pdf
```

المستخدم من جهاز آخر يصل للملف عبر NFS.

فكرة مهمة جدًا

النظام لا يسمح لأي شخص بالوصول لكل الملفات.

بل يحدد:

من يقرأ

من يكتب

من ينفذ





File Sharing – Multiple Users

- **User IDs** identify users, allowing permissions and protections to be per-user
- **Group IDs** allow users to be in groups, permitting group access rights

ما هو User ID؟

رقم أو معرف يميز كل مستخدم.

النظام يستخدمه لمعرفة:

من صاحب الملف

من لديه صلاحيات

ما هو Group ID؟

معرف لمجموعة مستخدمين.
الفكرة

بدل إعطاء صلاحيات لكل مستخدم منفصل:

يمكن إعطاء الصلاحيات لمجموعة كاملة.
مثال

لدينا مجموعة:

CS_Students

وتضم:

Ali

Sara

Omar

الملف:

project.pdf

يعطي صلاحية:

Read/Write for CS_Students

كل أعضاء المجموعة يحصلون على الصلاحية.

لماذا Groups مهمة؟

✅ أسهل بالإدارة

✅ تقلل التعقيد

✅ مناسبة للشركات والجامعات





File Sharing – Remote File Systems

الفكرة الأساسية
بدل مشاركة الملفات داخل نفس الجهاز فقط،
يمكن مشاركة الملفات بين أجهزة مختلفة عبر الشبكة.

الشبكة تُستخدم للوصول للملفات بين الأنظمة المختلفة.

- Uses networking to allow file system access between systems
  - Manually via programs like FTP
    يدويًا باستخدام برامج مثل: FTP
    اختصار: File Transfer Protocol
    يُستخدم لنقل الملفات بين الأجهزة.
  - Automatically, seamlessly using **distributed file systems**
    مشاركة تلقائية وشفافة.
    الملف البعيد يظهر:
    كأنه ملف محلي داخل جهازك.
    المستخدم لا يشعر أنه عبر الشبكة.
  - Semi automatically via the **world wide web**
    باستخدام:
    World Wide Web
    مثل: Google Drive, OneDrive, Dropbox
- **Client-server** model allows clients to mount remote file systems from servers
  يخزن الملفات Server
  يطلب الملفات Client
  - Server can serve multiple clients
    العميل يقوم بعمل: Mount
    لنظام ملفات موجود على السيرفر.
    ثم يظهر له كأنه جزء من جهازه.
    Server can serve multiple clients
    السيرفر يستطيع خدمة عدة أجهزة بنفس الوقت.
  - Client and user-on-client identification is insecure or complicated
    لأن النظام يحتاج التأكد:
    من هو المستخدم؟
    هل لديه صلاحية؟
    هل الاتصال آمن؟
    وهذا يصبح أصعب عبر الشبكة.
  - **NFS** is standard UNIX client-server file sharing protocol
  - **CIFS** is standard Windows protocol
  - Standard operating system file calls are translated into remote calls
    عندما البرنامج يطلب:
    open()
    read()
    write()
    النظام يحول هذه الأوامر إلى:
    طلبات عبر الشبكة.
- Distributed Information Systems **(distributed naming services)** such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing
  أنظمة تساعد على:
  إدارة المستخدمين
  الأسماء
  الأجهزة
  الصلاحيات
  في الأنظمة الموزعة.





File Sharing – Failure Modes

- Remote file systems add new failure modes, due to network failure, server failure
- Recovery from failure can involve state information about status of each remote request
- Stateless protocols such as NFS include all information in each request, allowing easy recovery but less security

المشكلة

الملفات البعيدة تعتمد على:
الشبكة
السيرفر

لذلك تظهر مشاكل جديدة.

أنواع الفشل

1) Network Failure

انقطاع الشبكة.

2) Server Failure

تعطل السيرفر.

المشكلة هنا

قد تكون العملية:
بدأت

لكن لم تكتمل.

مثل:

write file

ثم انقطع الاتصال.

Recovery
النظام يحتاج:
استرجاع الحالة بعد الفشل.

Stateful vs Stateless
Stateful
النظام يحتفظ بحالة كل طلب.

مثال:

من فتح الملف؟

أين وصل؟

Stateless
النص يقول:
NFS include all information in each request
المعنى
كل Request يحتوي كل المعلومات المطلوبة.
الميزة
سهولة التعافي من الأعطال.
العيب
أمان أقل وكفاءة أقل أحياناً.





Protection

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

الفكرة الأساسية

صاحب الملف يجب أن يتحكم:

ماذا يمكن فعله؟

ومن يستطيع فعله؟

أنواع الوصول (Access Types)

1) Read

قراءة الملف.

2) Write

تعديل الملف.

3) Execute

تشغيل الملف كبرنامج.

4) Append

إضافة بيانات في نهاية الملف فقط.

5) Delete

حذف الملف.

6) List

عرض محتويات المجلد.





Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users
  - a) **owner access** 7 ⇒ RWX = 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.

صلاحيات الوصول
النظام يستخدم:
r w x

ثلاث فئات من المستخدمين
1) Owner
صاحب الملف.
2) Group
مجموعة المستخدم.
3) Public / Others
باقي المستخدمين.

Owner Access = 7
111
يعني:
Read = 1
Write = 1
Execute = 1
إذاً:
7 = rwx
Group Access = 6
110
يعني:
Read
Write
بدون Execute
إذاً:
6 = rw-
Public Access = 1
001
يعني:
تنفيذ فقط
إذاً:
1 = --x





Windows XP Access-Control List Management

