THE UNIVERSITY *of* EDINBURGH
informatics

Operating Systems (INFR10079)

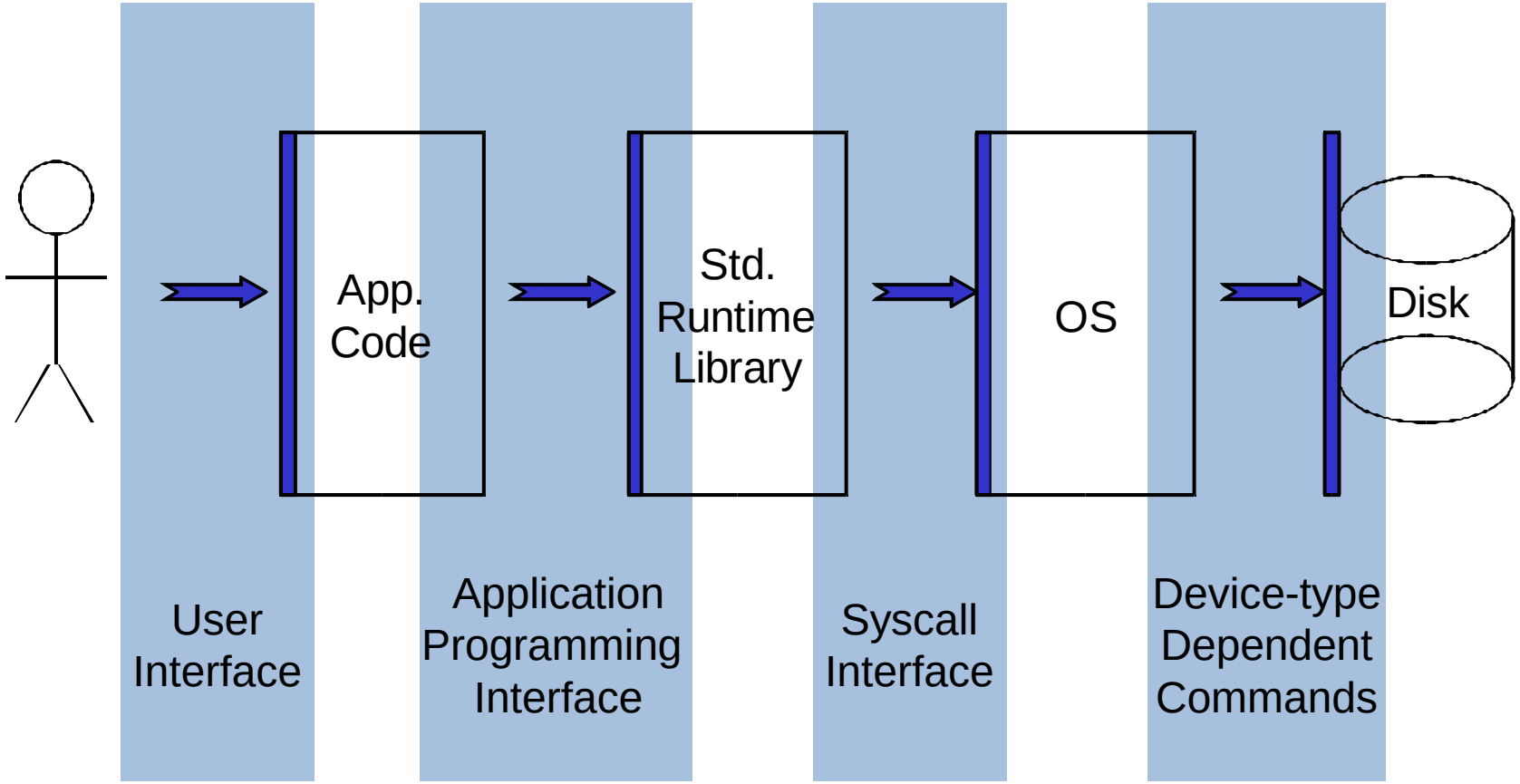
2022/2023 Semester 2

File System
(Basics)

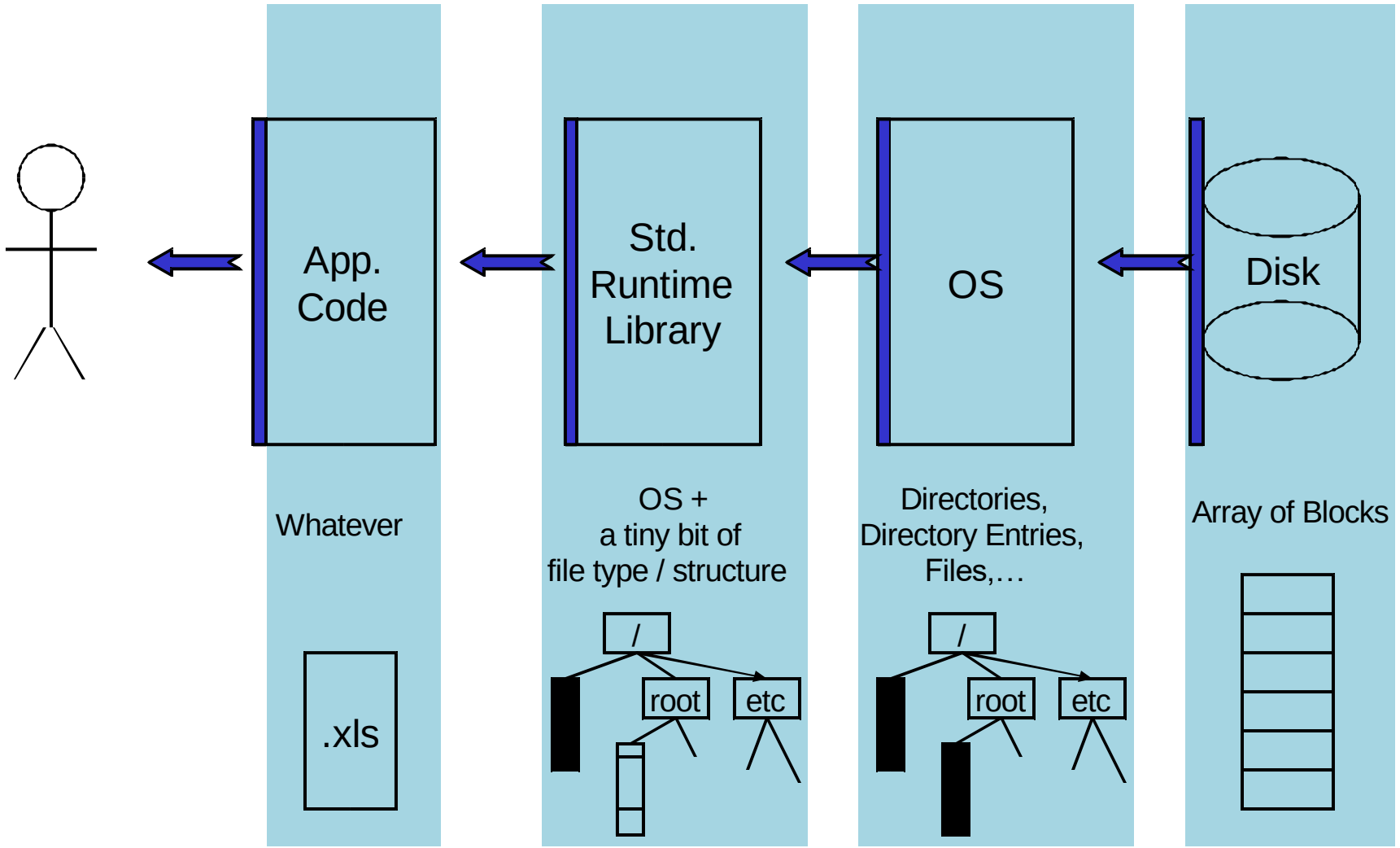
abarbala@inf.ed.ac.uk

Chapter 13, Chapter 14

Software/Hardware Interface Layers

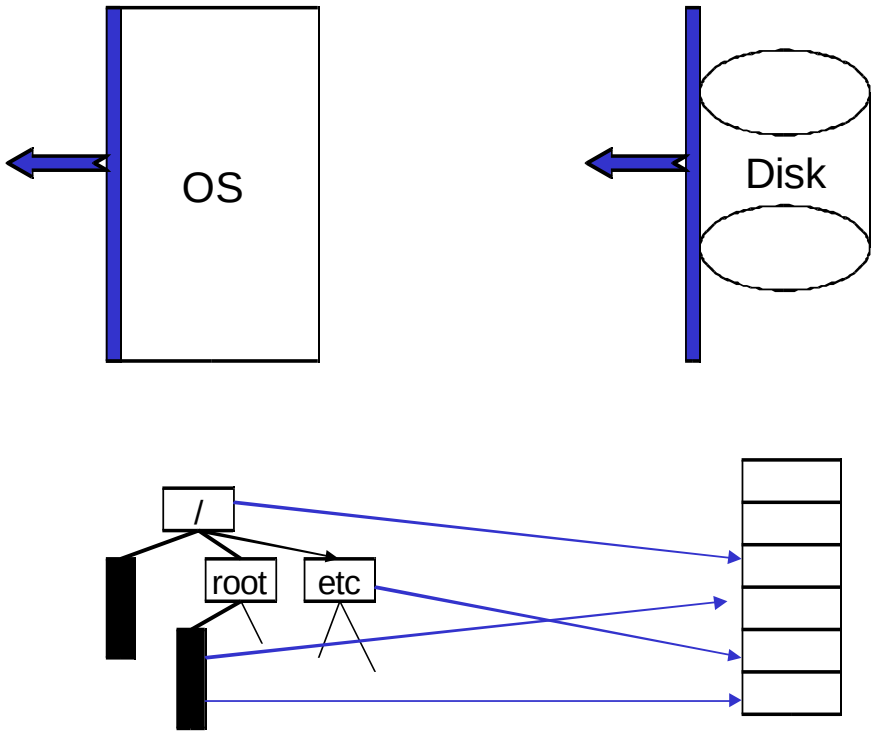


Software/Hardware
Exported Abstractions



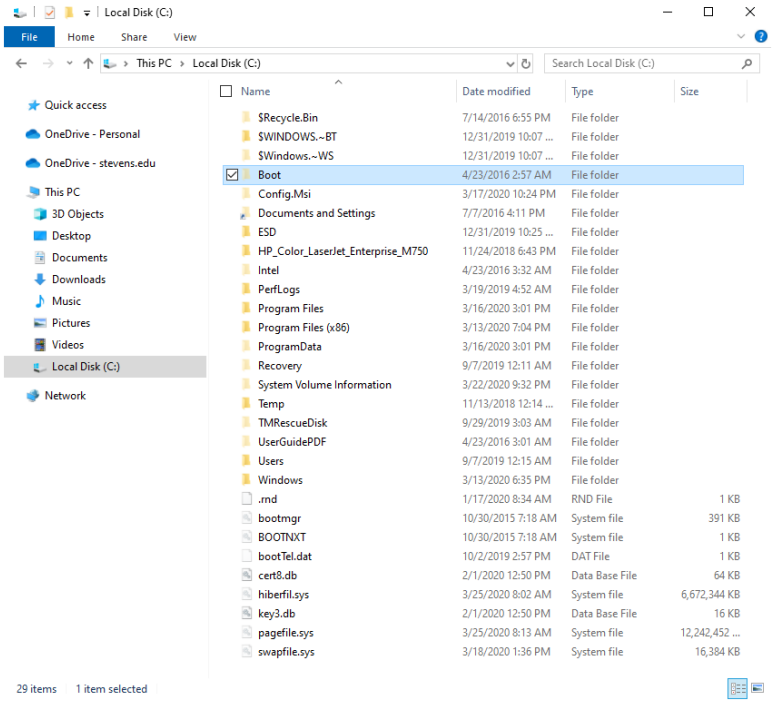
File System: Roles of the OS

- Hide hardware specifics
- Provide a uniform view
- Allocate disk blocks
- Access data
- Share data
- Check permissions
- Maintain metadata
- Performance
- Flexibility



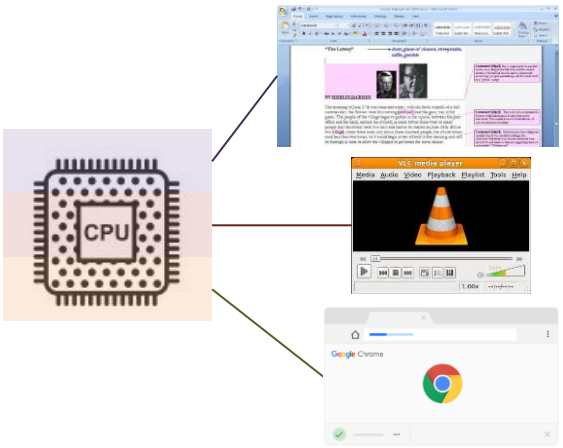
File System

- **Abstracts secondary storage**
  - Key abstraction are **files**
  - **Files** organized into **directories**
- Enables **sharing** of data between
  - Processes
  - People
  - Machines
  - etc.
- Provides additional
  - Access control
  - Consistency
  - Reliability
  - etc.

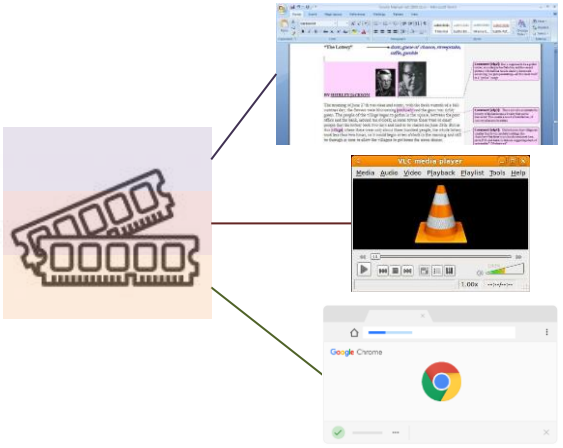


File #1

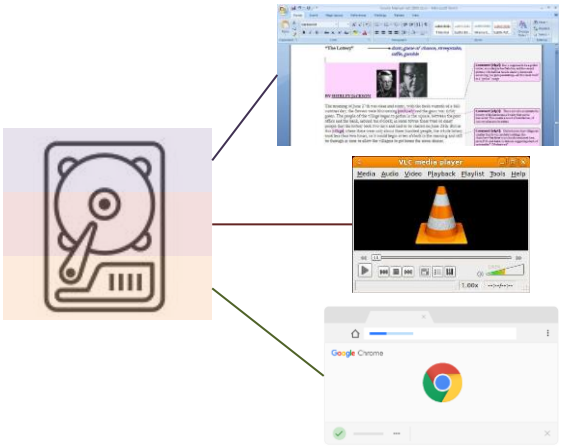
- A file is an abstraction
  - The OS **abstracts away** the concept of disk to offer files
  - **Shield the user** from the details about storage



Process, abstracts physical CPU

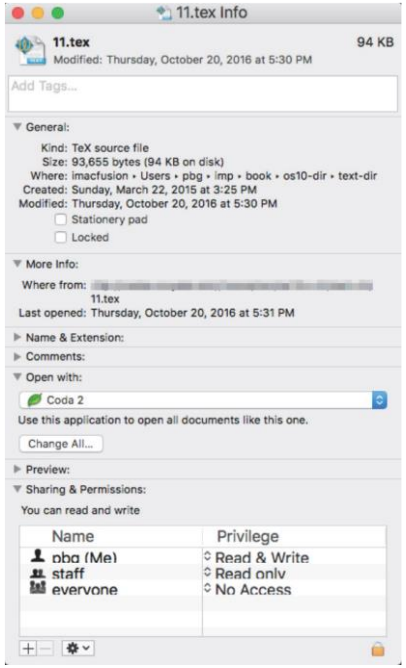


Address space, abstracts physical memory



File, abstracts disk

File #2



- A **named collection of related information** with some properties
  - Content, size, owner, protection, last read/write time …
- Files **types**
  - Understood by **file system**
    - **Directory, symbolic link, devices**
  - Understood by **other parts of OS, libraries, application**
    - **Programs**: executable, object code, source code
    - **Data**: numeric, alphabetic, alphanumeric, binary
- **Type** can be **encoded** in the file's name or content
  - Windows encodes types in name
    - .com, .exe, .bat, .dll, .jpg, .mov, .mp3, …
  - Linux deducts the type from the content

Basic Operations

Unix

- create(name)
- open(name, mode)
- read(fd, buf, len)
- write(fd, buf, len)
- sync(fd)
- seek(fd, pos)
- close(fd)
- remove(name)
- rename(old, new)

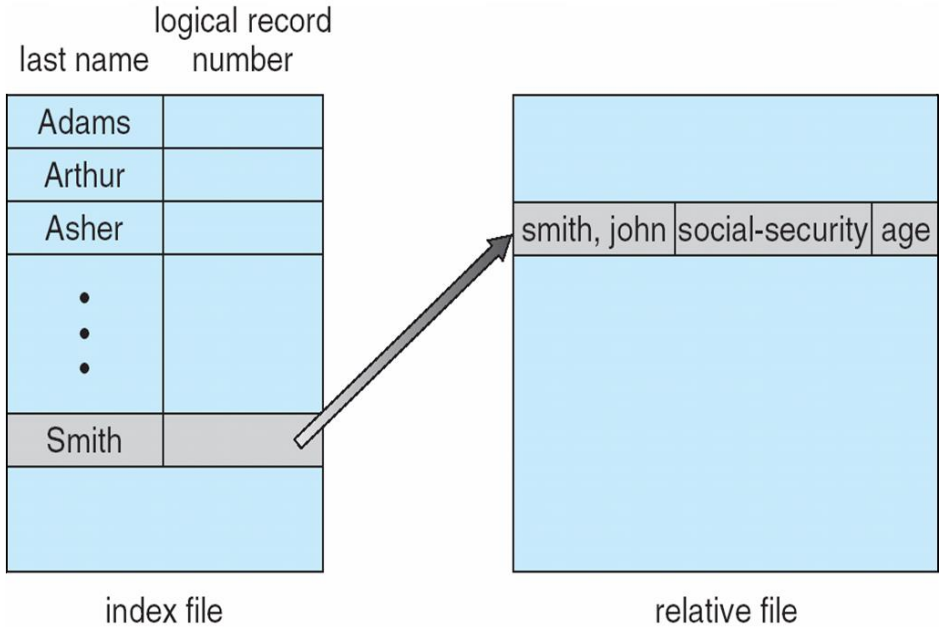
Windows

- CreateFile(name, CREATE)
- CreateFile(name, OPEN)
- ReadFile(handle, …)
- WriteFile(handle, …)
- FlushFileBuffers(handle, …)
- SetFilePointer(handle, …)
- CloseHandle(handle, …)
- DeleteFile(name)
- MoveFile(name)
- CopyFile(name)

Application Programming Interface

File Access Methods

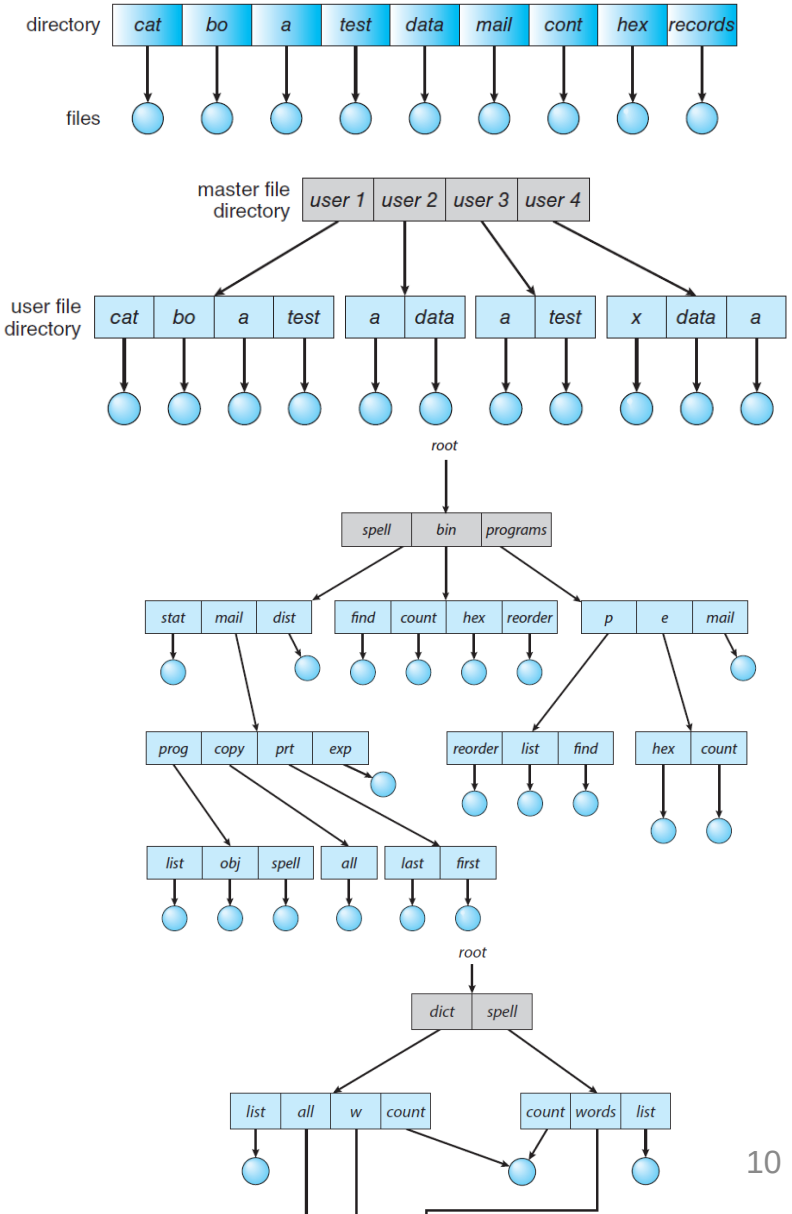
- File systems provide different access methods
  - **Sequential**
    - Read/write bytes one at a time, in order
  - **Direct**
    - Random access given a byte #
  - **Record**
    - File is an array of fixed- or variable-sized records
  - **Indexed**
    - One file contains an index to a record in another file



Indexed Access
(e.g., Database)

Directories Stuctures

- **Single-level** directory
  - File must have unique names
- **Two-level** directory (per-user directory)
  - Sharing requires introduction of path abstraction
- **Tree structured** directories
  - Eventual replication of files
- **Acyclic-graph** directories
  - Links as a solution



Directories

- Directories provide
  - Way for users to **organize their files**
  - Convenient **file name space** for user and FS

- Most file systems support **multi-level directories**
  - Naming hierarchies (/, /usr, /usr/local, /usr/local/bin, …)
- Most file systems support the notion of **current directory**

- **Absolute names:** fully-qualified starting from root of FS

```
bash$ cd /usr/local
```

- **Relative names:** specified with respect to current directory

```
bash$ cd /usr/local   (absolute)
bash$ cd bin          (relative, equivalent to cd /usr/local/bin)
```

Directory Internals

- Directory **is typically just a file** that happens to contain special metadata
- Organized as a **symbol table**
  - List of <name of file, reference to file>
  - Hash table of <name of file, reference to file>

- **Attributes** include such things as
  - Size, protection, location on disk, creation time, access time, …

- The directory list is usually **unordered**
  - When you type “ls”, the “ls” command sorts the results for you

Path Name Translation Example

- You want to open “/one/two/three”

```
fd = open(“/one/two/three”, O_RDWR);
```

- Inside the file system
  1. Open directory “/” (well known, can always find)
  2. Search the directory for “one”, get location of “one”
  3. Open directory “one”, search for “two”, get location of “two”
  4. Open directory “two”, search for “three”, get location of “three”
  5. Open file “three”
  - Of course, permissions are checked at each step
- FS spends much time walking down directory paths
  - OS will cache prefix lookups to enhance performance
    - /a/b, /a/bb, /a/bbb all share the “/a” prefix

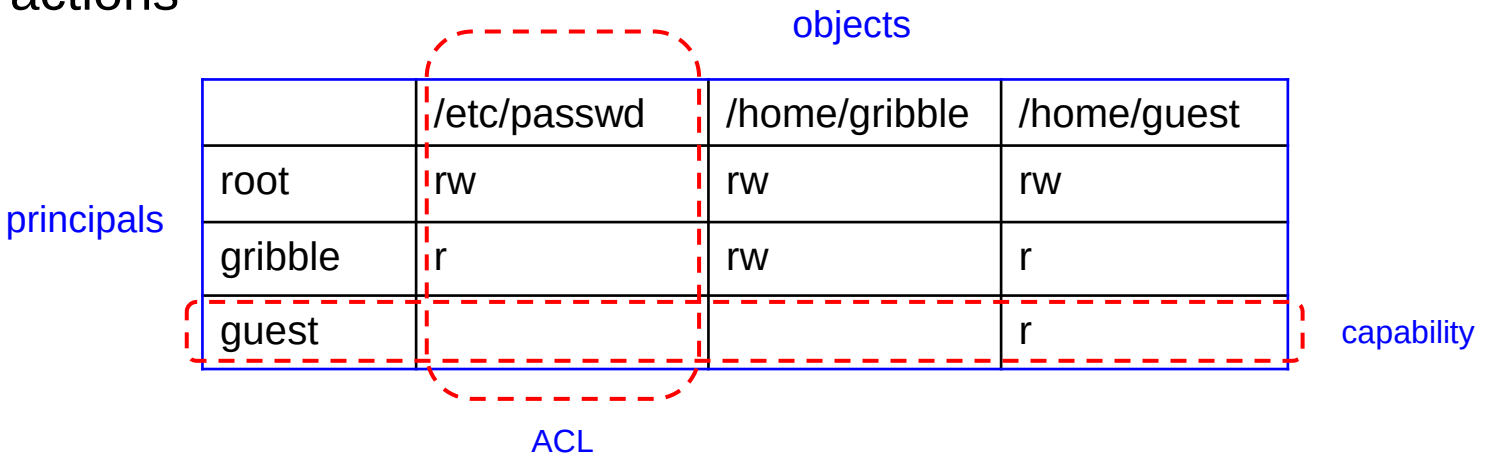
File Protection

- File System implements a protection system
  - Control **who** (user) can access **what** (file)
  - Control **how** the file can be accessed by user (e.g., read, write, or exec)
- Often generalized
  - Generalize files to objects (the “what”)
  - Generalize users to principals (the “who”, user or program)
  - Generalize read/write to actions (the “how”, or operations)

- Protection system dictates whether a given action performed by a given principal on a given object should be allowed
  - E.g., you can read or write your files, but others cannot
  - E.g., your can read `/group/teaching/cs3` but you cannot write to it

Protection Models

- Two different models
  - Access Control Lists (ACLs)
    - For each **object, keep list of principals** and principals' allowed actions
  - Capabilities
    - For each **principal, keep list of objects** and principal's allowed actions



- Condense the length of the **ACL** by using **three class of users**
  - Owner
  - Group
  - Other