OS I: Week-2

OS:

Web server program -> Operating Systems -> hardware/storage

Kernal controls the hardware directly, provides services/resoruces for applications to run and interact with the users. Also manages the security/manage access.

Low-level software = @ hardware layer

Top layers - user space, includes apps and services (Programs running behind the scenes).

Shell: Original parts for systems that dont have a user interface

GUI- runs shell programs behind the scenes.

Interfaces:

Servers may or may not have something serving an interface

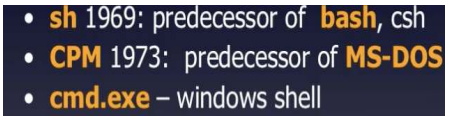
Psychology - Human-Computer-Interaction

- Requires multiples iterations to get it right

GUI vs CLI (Command-Line-Interface):

CLI only interacts through a keyboard and monitor (which only prints text)

- First type of interfaces for computers



GUI - Responsive, use of windows, icons, menu and pointer/touch device (WIMP interface)

Both have CONS and PROS, each is IMPORTANT and APPROPRIATE.

- Can be COMBINED aswell, for example: Moving from GUI to a Shell prgm

Use of multiple interfaces

- Tool box IF: To change interface to suit your need. e.g

- Automation of repetitive tasks
- Understanding to use multiple interfaces

CLI PROS

- Flexible - Combine commands
- Fine tuning -> Parameters
- Essential for system administration
- Faster in comparison to GUI and runs on simple hardware
- Can run offline or remotely
- Robust - Hard to crash

CLI Weaknesses:

- Hard to learn - Cryptic commands and parameters
- Multiple options in performing commands
- Output often cryptic or non-existent
- Inconsistent commands - Different version of UNIX, but different. DOS, completely different command between versions
- No safety net - You need to know exactly what to write

Scripting:

Batch file can automate CLI: /bin/bash used in unix version

- ls -dl r* (ls: lists files, dl: displays directories, r*: everything that starts with r)

```
#!/bin/bash
#
echo "now executing ls -dl ${1}"
ls -dl ${1}
```

Putting that command into a file called list.sh, CLI will treat this file as a command if we set certain parameters, in this case make it executable.

#!/bin/bash the command that executes the bash file

echo displays what the parameters are

r* replaced by ${1}, means the first parameter to the list.sh command/file.

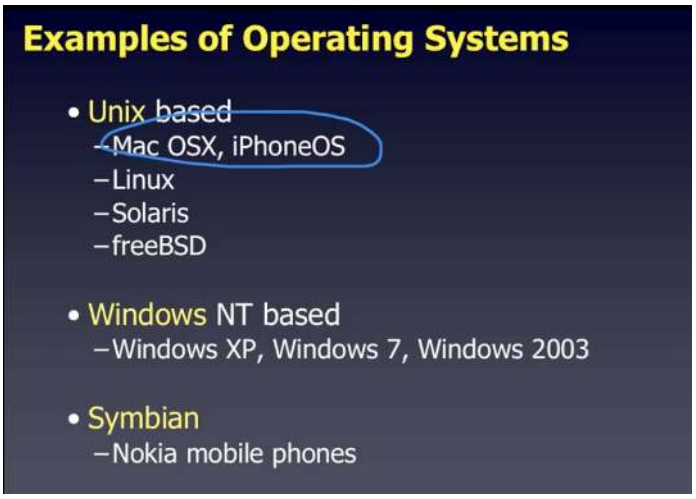
Can reuse the command/file by giving a different parameter. E.g List.sh d*

Some GUI's also have batch facilities called "macro". This type of programming language is called a "Scripting language".

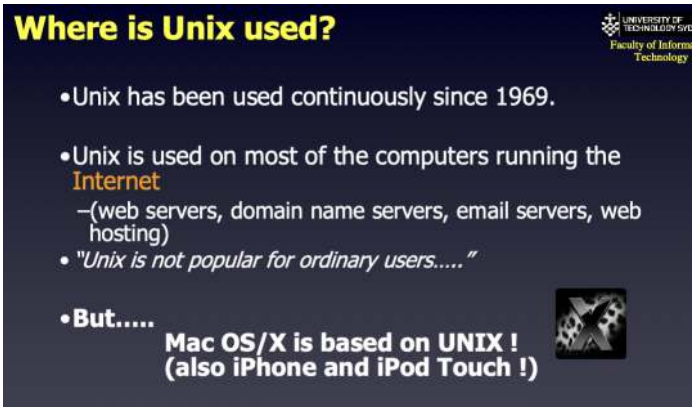
Characteristics:

- One key characteristic of most scripting languages is that they typically have loosely bound variables. Same variable can be used as a number or a string.
- Language syntax is often inconsistent
- Usually run through an interpreter, not a compiler. - Means a program reads the actual operations in the file, then executes them.

OS II: Week-3



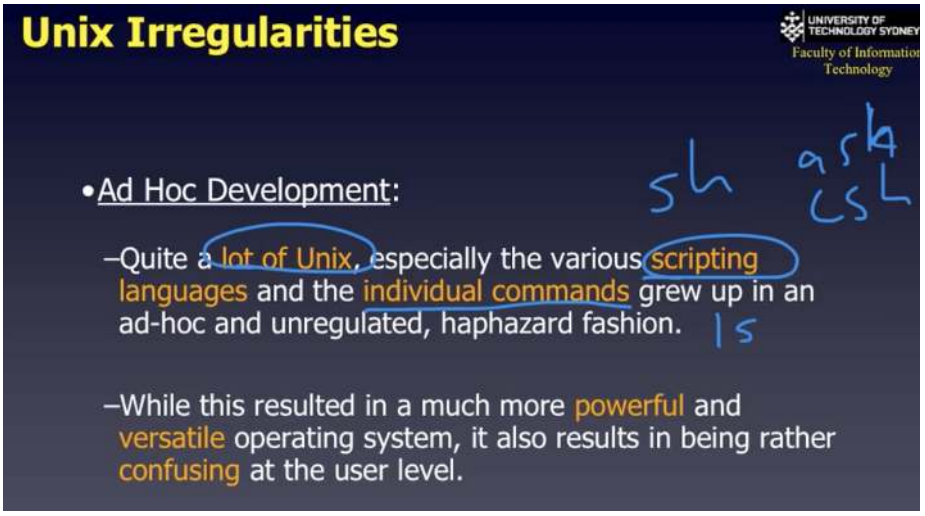
Unix:



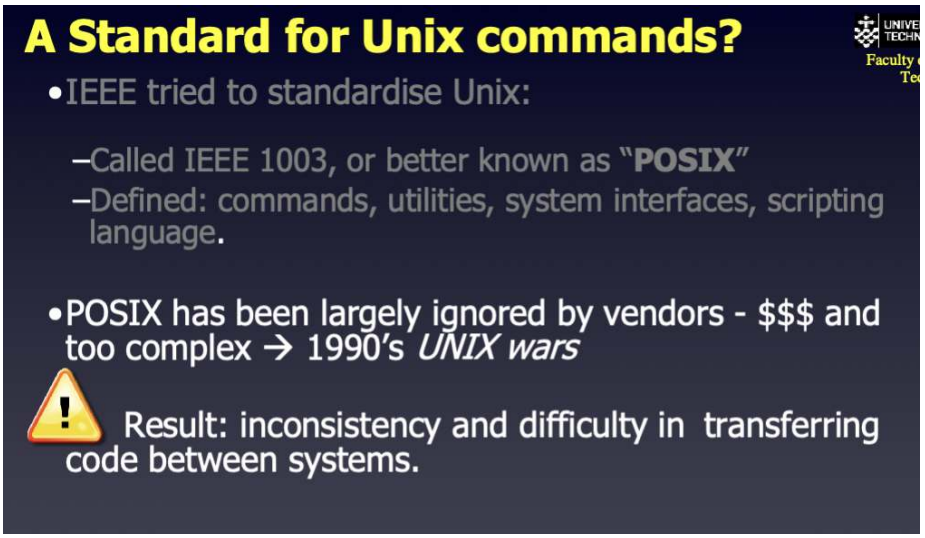
Started as a research project, BSD developed its own version which resulted in Darwin(Basic kernel) used in Apple OSX (IOS & Mac).

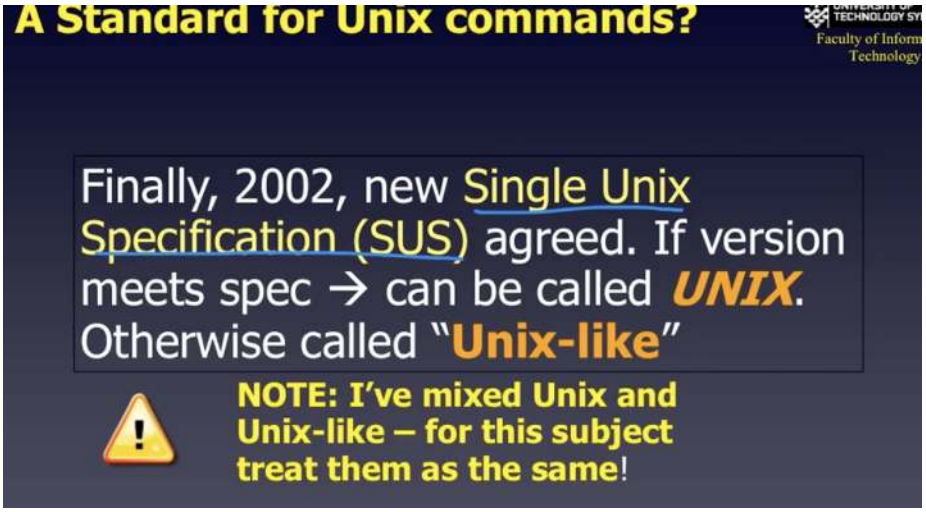
Microsoft started a version aswell.

GNU as well which resulted in Linux, this version was made separately and was open-source. Linux resulted from GNU and Minux version.



Things get confused, which command option do I use etc.





Why has Unix survived? (1)

UNIVERSITY OF TECHNOLOGY SYD
Faculty of Informa Technology

No one owns these ideas.

- **Unix is a set of ideas**, none of which are secret.
- **Any person or group is free to implement these ideas**. There have been court cases over specific lines of code in "official" Unix (System V) , but the lines of code are only a specific implementation of these principles.

Never patented, based on simple concepts

Unix is based on simple concepts:

i.e.: Files, processes, permissions and users.

- Even hardware devices e.g. /dev/mouse are represented as files.
- This has simplified the conceptual picture of Unix (if not the internal code)
- It has also allowed Unix to incorporate new ideas and technologies quite easily.

Unix is efficient, stable and relatively secure

- Unix is **fast** and **stable** (system crashes are rare).
- Designed for **security for multi-user systems** – files have owners, security permissions are tight
  → therefore fewer viruses for Unix.

More stable than windows

The Unix as a set of tools approach

- The Unix CLI has some very powerful features. Specifically, **simple commands, pipes and I/O redirection**.
- You can create very powerful ad hoc tools → by passing the output of one command to another command
- → This has a great appeal to many technically oriented users.

Filesystem:

What is an Operating System?

- Manages your computer
- Runs programs
- Interface between user and hardware
- Provides services to programs & users
- Protects users and programs from each other....
- Manages resources e.g. File System

File systems

A file system is a part of the operating system that manages data storage and access.

Classified into:

- Logical File System
- Physical File System

File systems

- Logical file system
  - How we view the file system
    - files
    - directories/subdirectories
    - partitions
- Physical file system
  - How these items are physically represented and stored

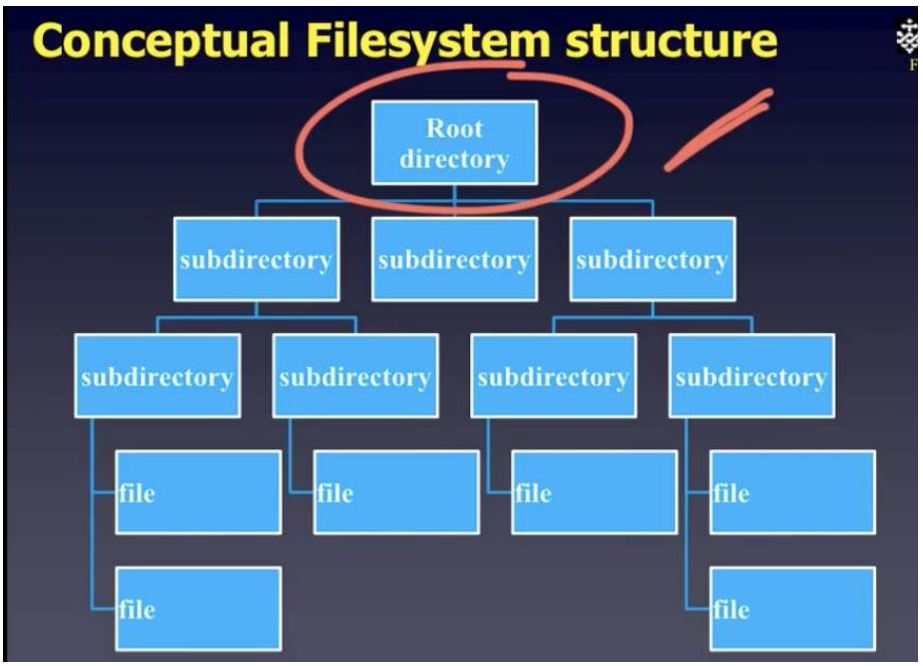
Logical - What you see

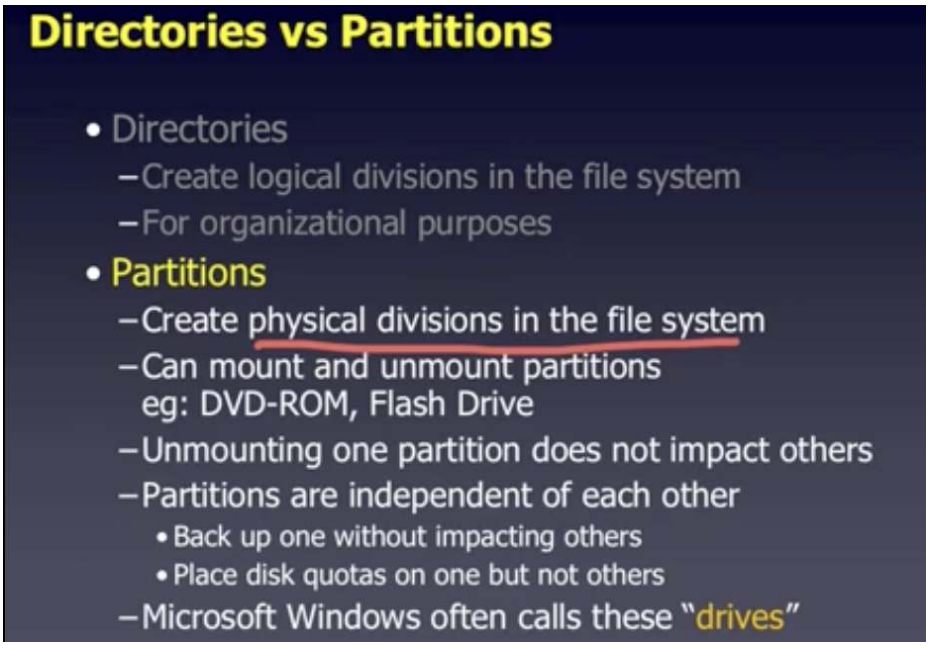
Internaly, its stored in a physical file system

Logical File System

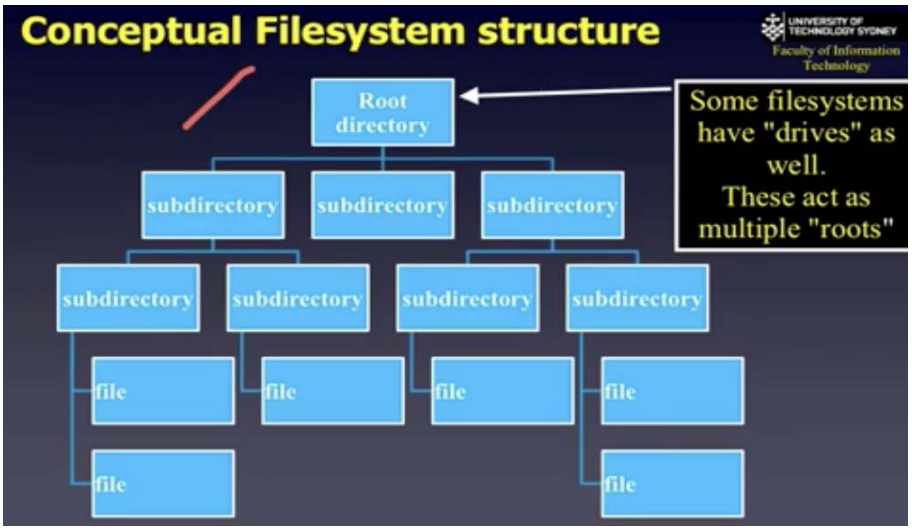
- Files
  - Executable files (programs)
  - Data files
- Directories
  - Store files and (usually) subdirectories
  - Often hierarchical ("tree") format
- Partitions
  - Some directories may reside in different partitions from other directories
  - Abstracts physical infrastructure from users
  -

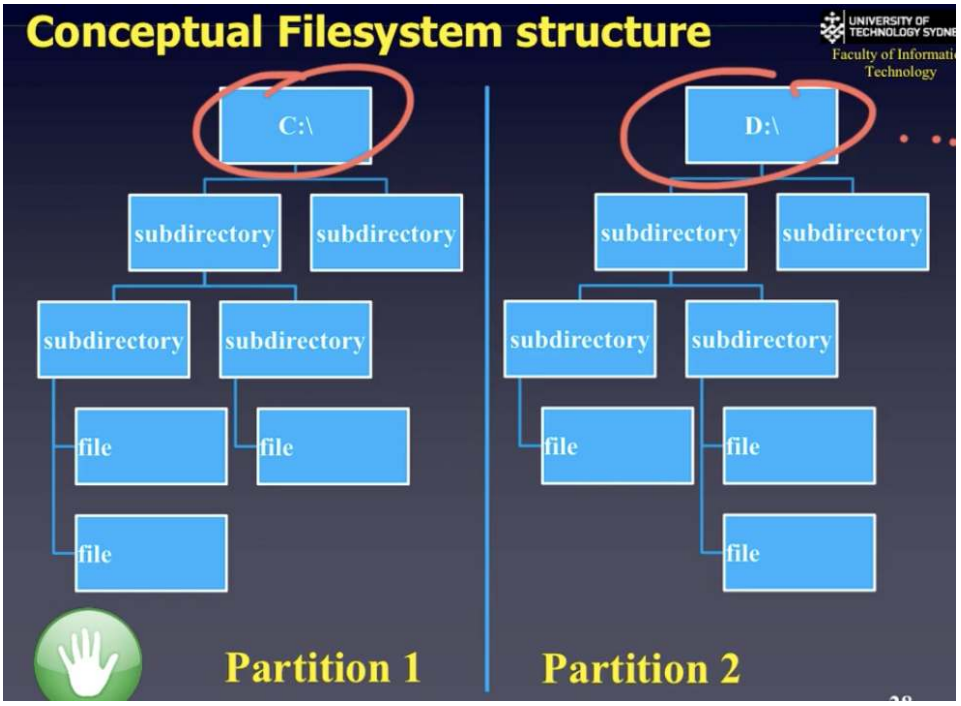
Files- HTML, gifs, jpegs, word docs
physical infrastructure - Partitions- Drives, CDs, disks, USB



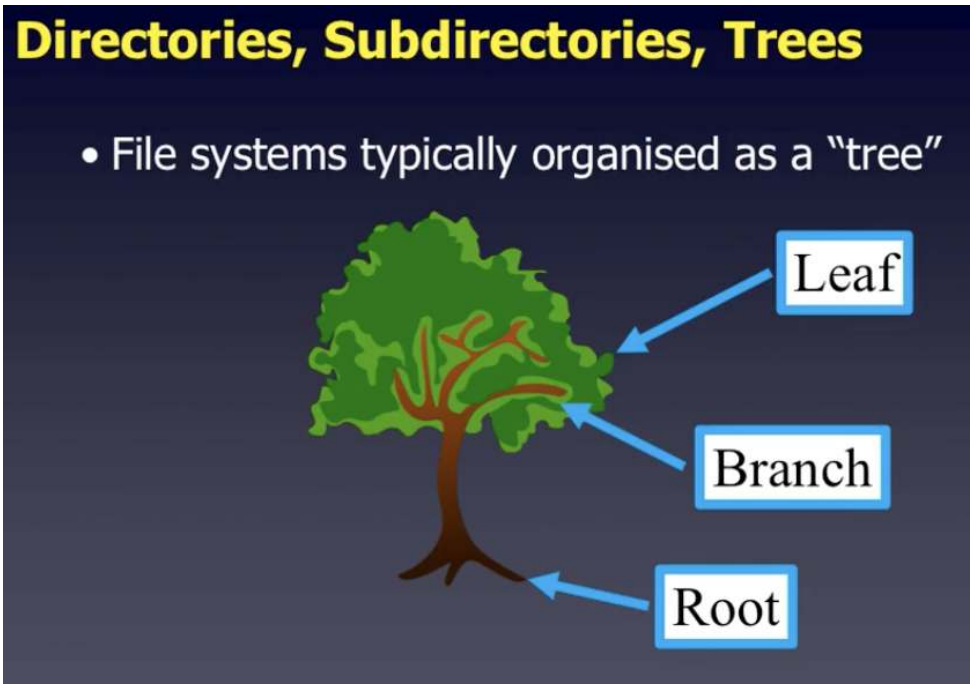


Partitions do not impact root directory

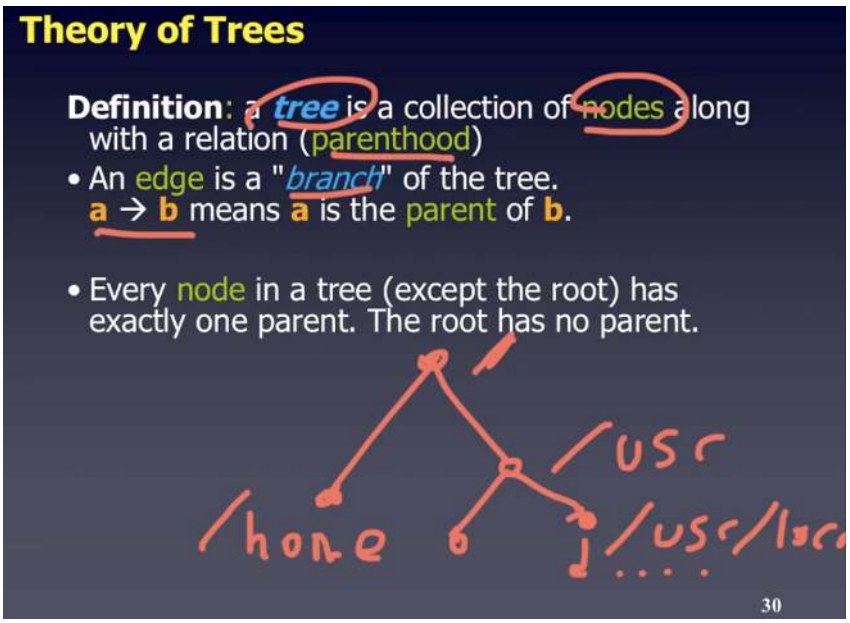




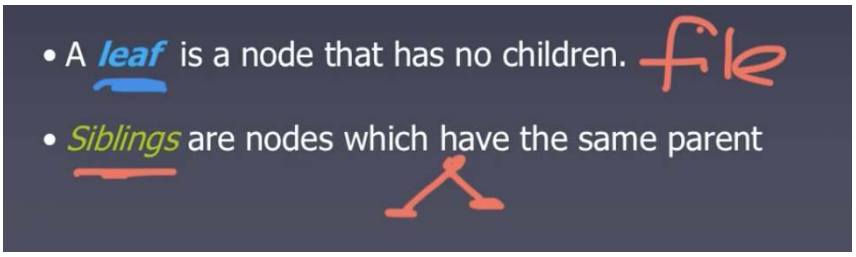
Structured logically like any other filesystem

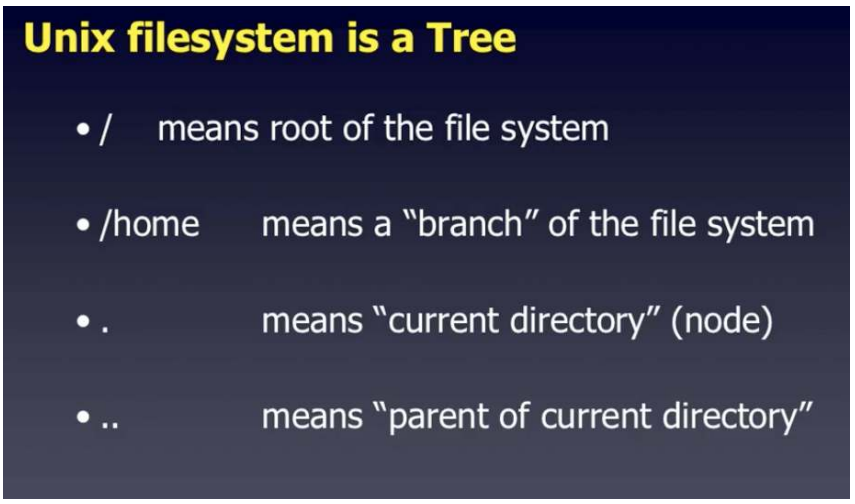


Roots at the top in the context of computer science



Edge is the branch, node is the file, parenthood relates to the relation between the root -> Edge -> Node



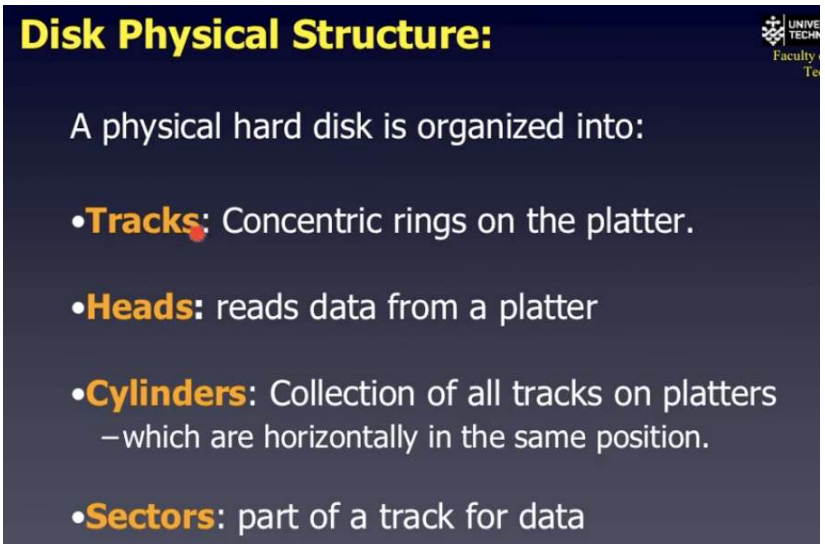


File storage

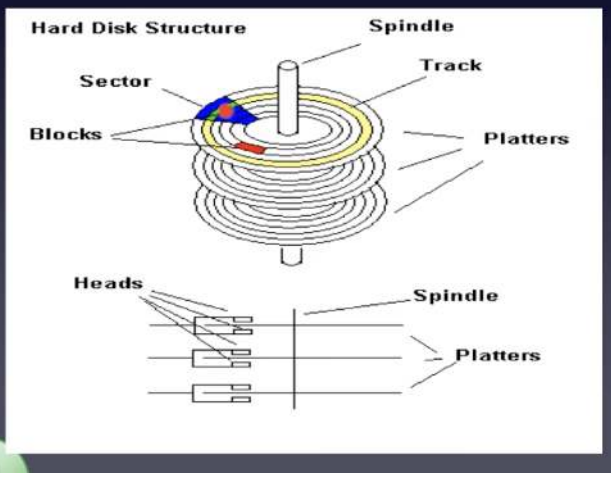
This is where most files are stored

File systems and file manipulation

Hard disk and SSD is managed and organised by an operating system comprising of; Disk physical structure, Disk logical structures and File allocation methods



Low Level Hard-Disk Data Storage



Disk Structure

- A disk is a stack of magnetic platters.
  - This stack is divided into cylinders
  - each cylinder contains circular tracks
    - which are in turn divided into sectors.
- Read/write operations are provided by the disk heads
  - These move concurrently along the fixed disk arm.
- The disk itself rotates with constant angular velocity to provide access to every sector.

That's obsolete and everything will eventually be replaced with SSDs (PCI, NVMe)

Disk Formatting

- Formatting is the operation which creates the **physical** disk structure.
- Formatting is organizing and marking the surface of a disk into tracks, sectors , and cylinders.

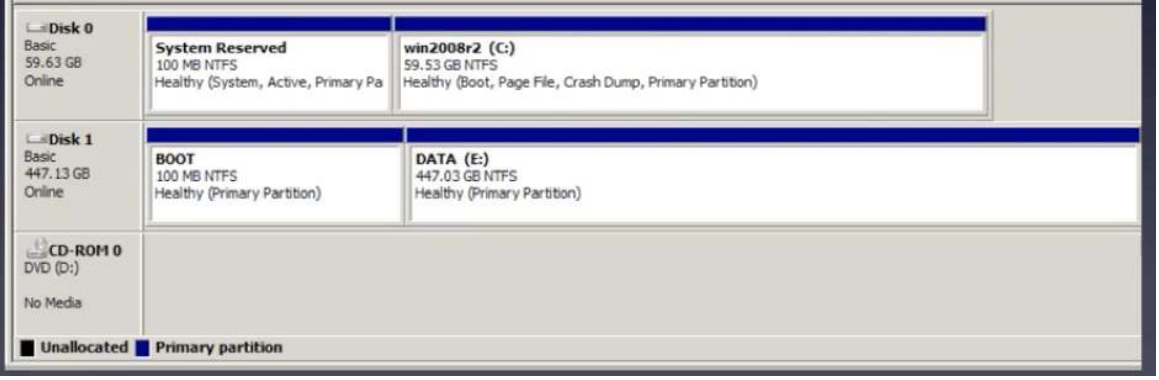
Done at a hardware level, done by hand.

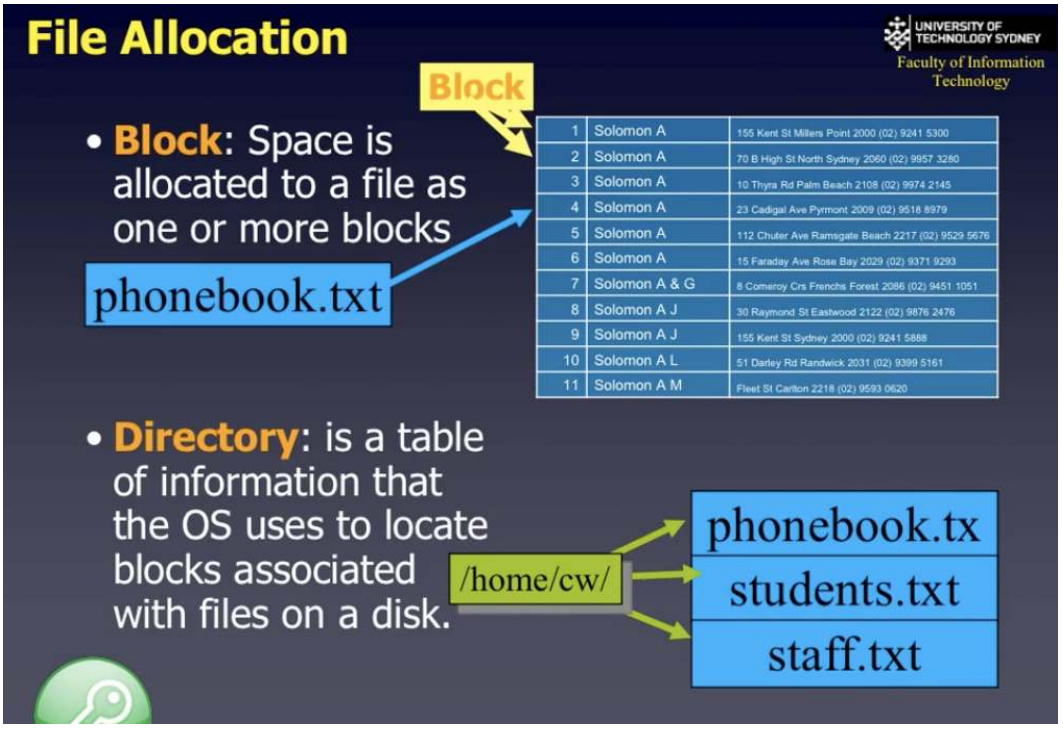
- **Partitions**: Disks can be subdivided into partitions
  – each is an independent storage device.

- **Blocks**: The operating system views all the disk space as an array of fixed size logical blocks.
  – A logical block is the smallest unit of data to transfer.

Example: Windows Partitions

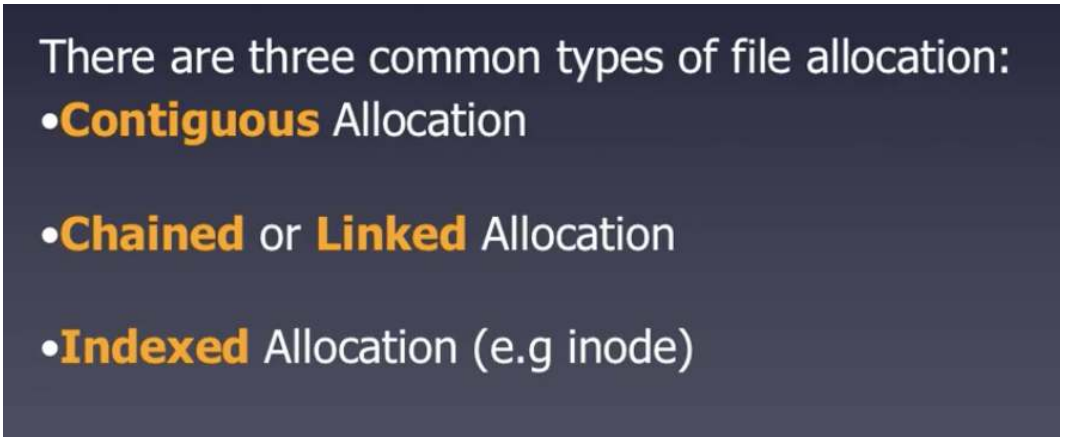
Volume	Layout	Type	File System	Status	Capacity	Free Space	% Free	Fault Tolerance	Overhead
BOOT	Simple	Basic	NTFS	Healthy (Primary Partition)	100 MB	84 MB	84 %	No	0%
DATA (E:)	Simple	Basic	NTFS	Healthy (Primary Partition)	447.03 GB	320.42 GB	72 %	No	0%
System Reserved	Simple	Basic	NTFS	Healthy (System, Active, Primary Partition)	100 MB	71 MB	71 %	No	0%
win2008r2 (C:)	Simple	Basic	NTFS	Healthy (Boot, Page File, Crash Dump, Primary Partition)	59.53 GB	21.41 GB	36 %	No	0%





Each row is stored in a block

Directory is a special type of file used to locate blocks associated with files on a disk



File Storage - Contiguous

- A single contiguous set of blocks is allocated to a file at the time of file creation.

03	lo	n	m	Th	a	Pa	Be	21	(02	97	21
So	mo	A	10	yr	rd	lm	ac	08	)9	4-	45

1 2 3 4 5 6 7 8 9 10 11 12 13

- To access information in block B, this information resides at block number starting block + B
- Supports random access: you know exactly where every block is after the starting block.
- Fragmentation of unused space (external fragmentation) will occur, needs compaction.
- Often used in magnetic tapes rather than disks

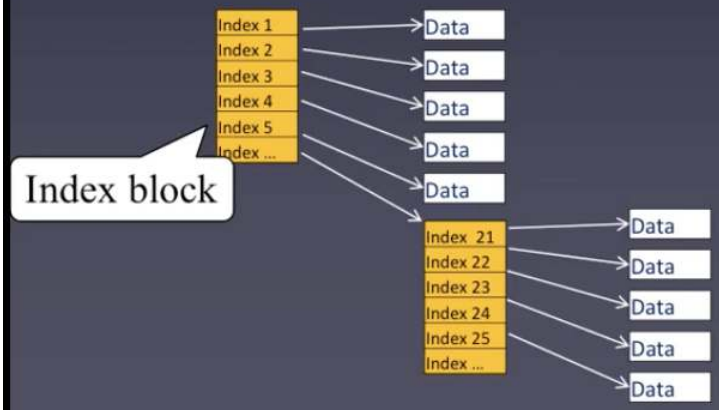
Downside to this is fragmentation

Chained Allocation

- File is written as a collection of non-contiguous blocks
- File is implemented as a linked list of blocks
- Each block contains a (pointer to) the address of next block.
  - Last block contains invalid (negative) number (End-Of-File marker)
- Directory entry contains the head (starting) block number and length of the file
- Chained is good for sequential access, bad for random access

Indexed Allocation

- ➔ "Tree" based allocation system
- A special "index" data block will contain a list of data blocks# for the file
- If the file is too big, the "index" data block will point to other "index" data blocks

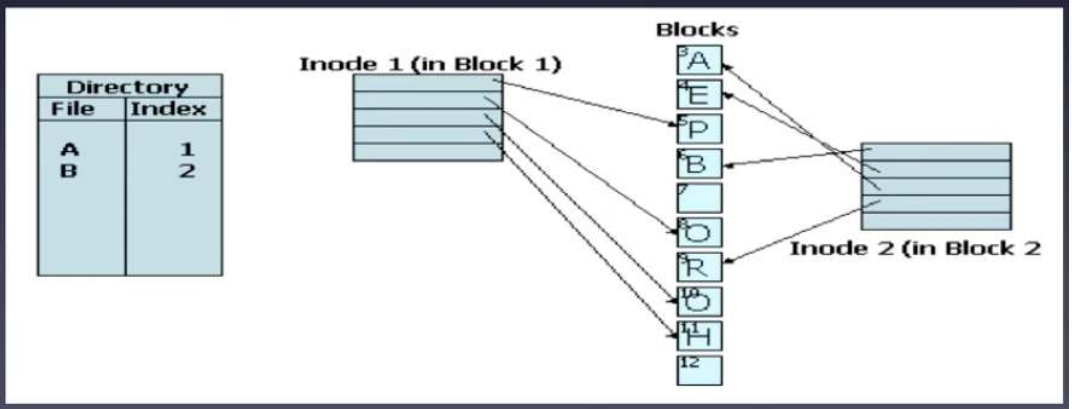


Second or Indirect block that points to more blocks. Expandable file allocation system

File storage: Indexed – UNIX

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Information Technology

- On UNIX, files are stored on blocks called an **inode**
- Each file/directory is referenced by an inode



- Very efficient use of space and fast to read blocks

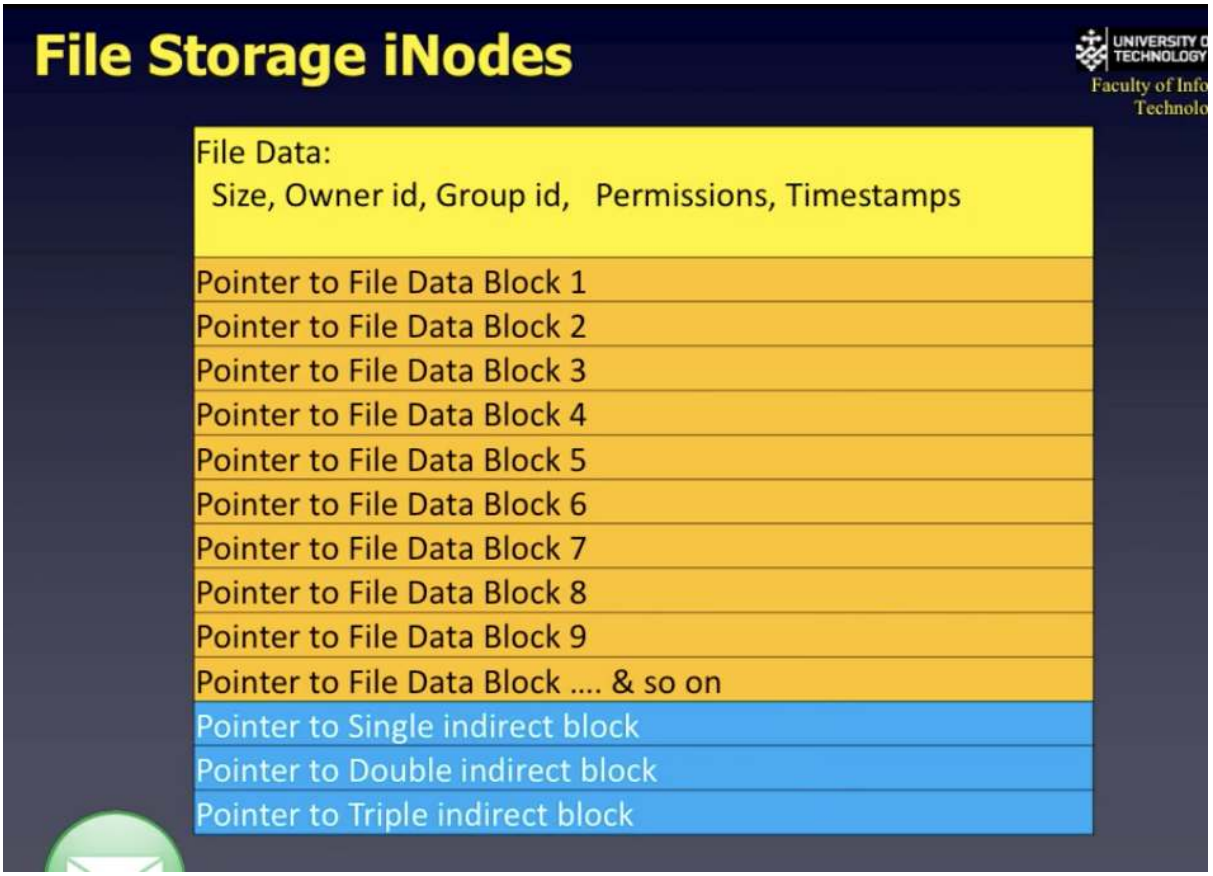
iNodes and Directories in Unix

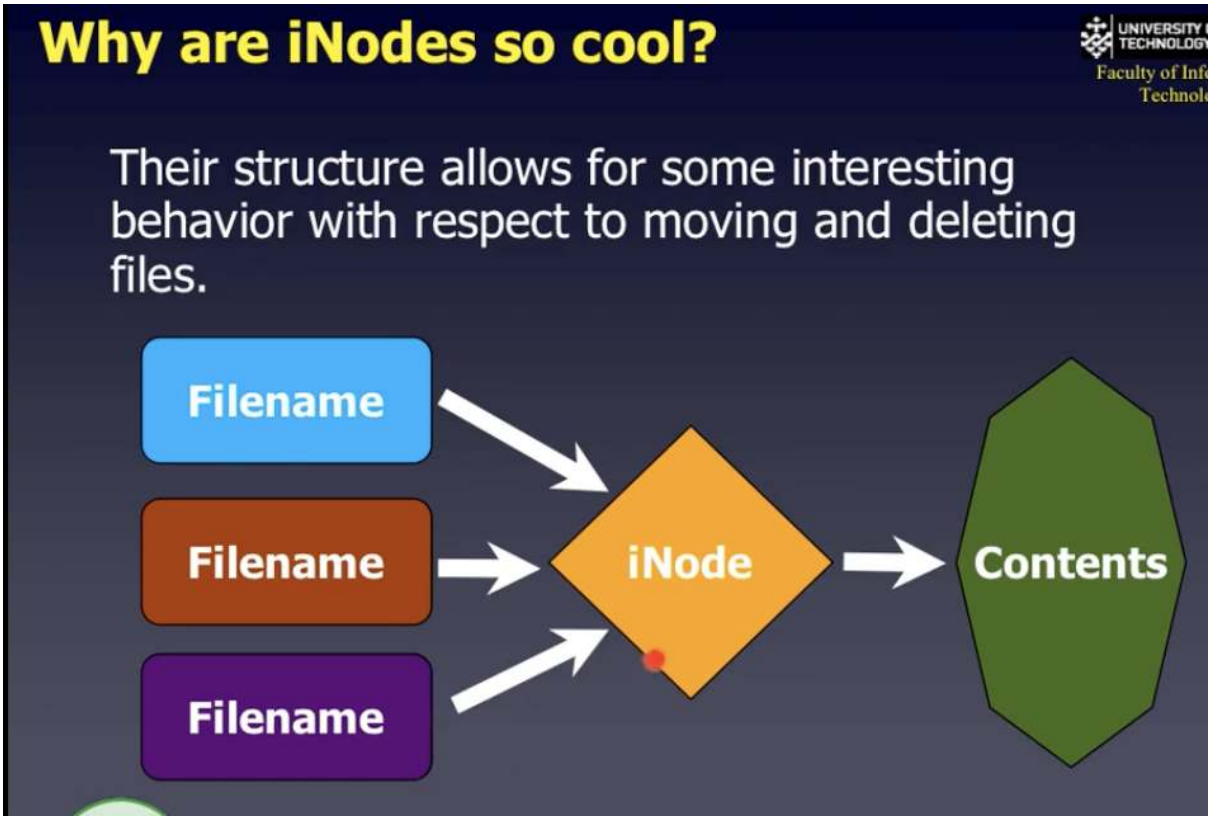
- The inode system is used in unix.
- All inodes are numbered.
- Special blocks/file on disk called a directory
- Directories contain the names of files and the inode number for the file.
- Notice that the inode structure is a **tree**

What does an inode store?

- File Metadata: Size, Owner id, Group id, Permissions, Timestamps
  → *NOTE: inode does NOT store name of file*!!!
- Pointers to the blocks that store the files data

- (optionally)
  - "Single indirect block"
    → pointer to a disk block → contains an index of pointers to data blocks
  - "double indirect block"
    → Pointer to more "single indirect blocks"
  - "triple indirect block"
    → pointer to more "double indirect blocks"

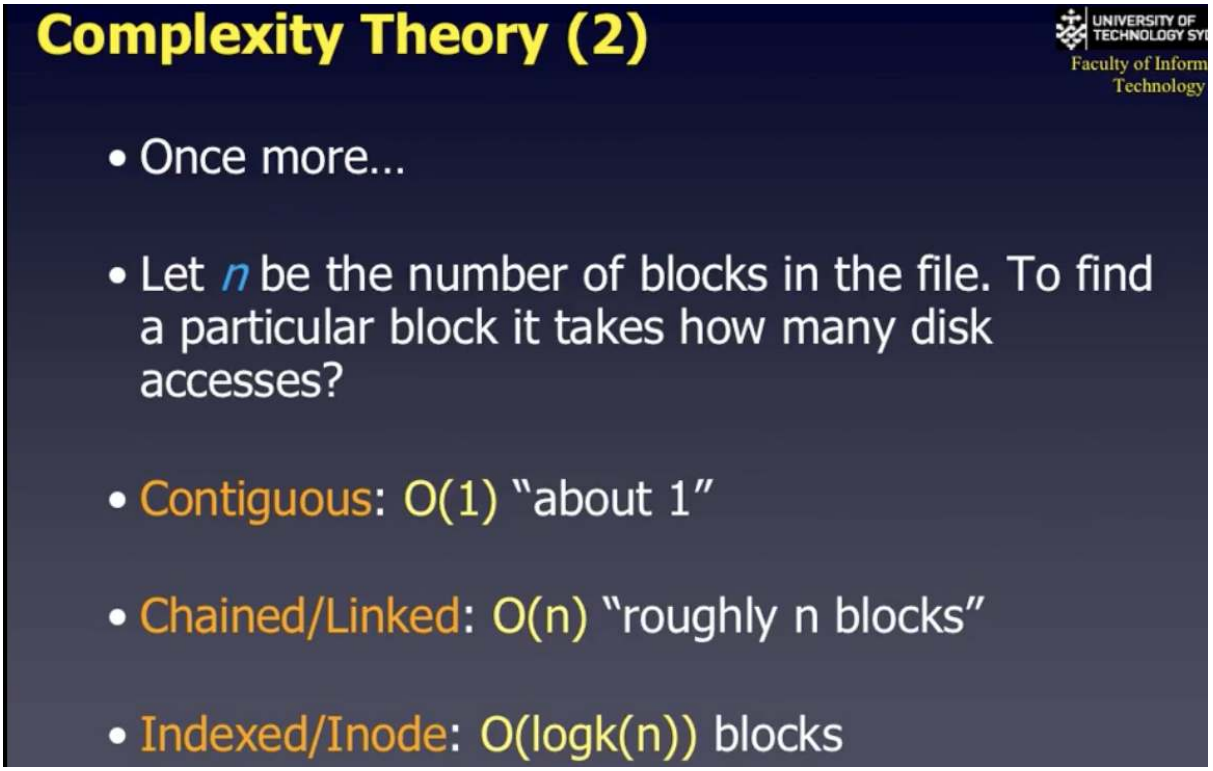




Different filenames pointing to the same inode

- Contiguous is great for "direct" storage & tiny file systems
- Chained/Linked is good for archival (eg backup)
- Indexed is the only reasonable option for large systems

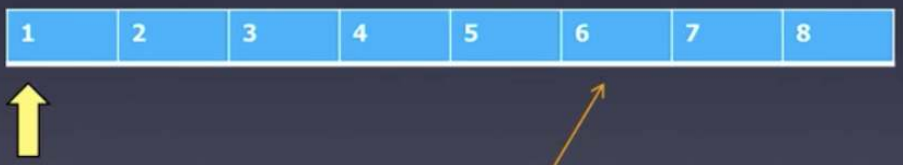
Complexity



How many reads = how complex

Complexity Theory - explanation

- Chained O(1):
  - this is constant # i.e to read ANY block
    1. Calculate block to read = Starting block + block# -1
    2. Read the block directly ie: we only have 1 read for ANY block in the file



Starting block

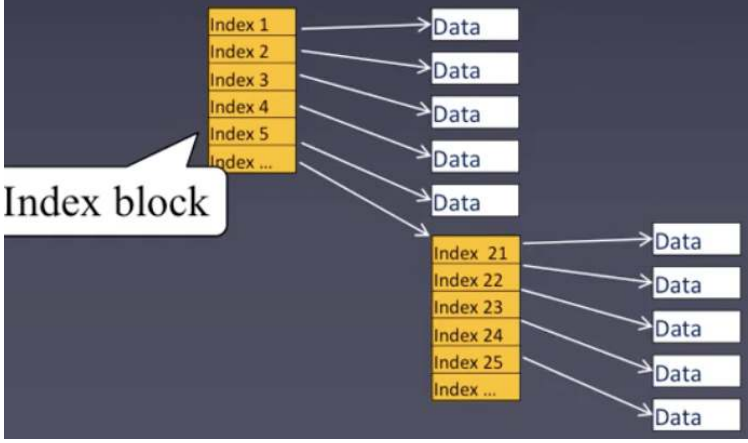
- I want to read block #6?
  - ➔ Calculate: block# = starting block + 6 -1
    = 1 + 6 -1
    = 6
  - ➔ solution: read block 6 !

- Linked O(n):
  - this is proportional # i.e to read ANY block
    1. Start at "Starting block" + block# -1
    2. Read each block sequentially until you have reached the block# required
       - Best case: read block#1: "starting block" is it! ➔ 1 read
       - Worst case: read block n (where n = last block)
         ➔ read 1st, then 2nd, then 3rd .... Then (n-1)th then nth block
       - Average case: halfway ie: n/2

tarting block

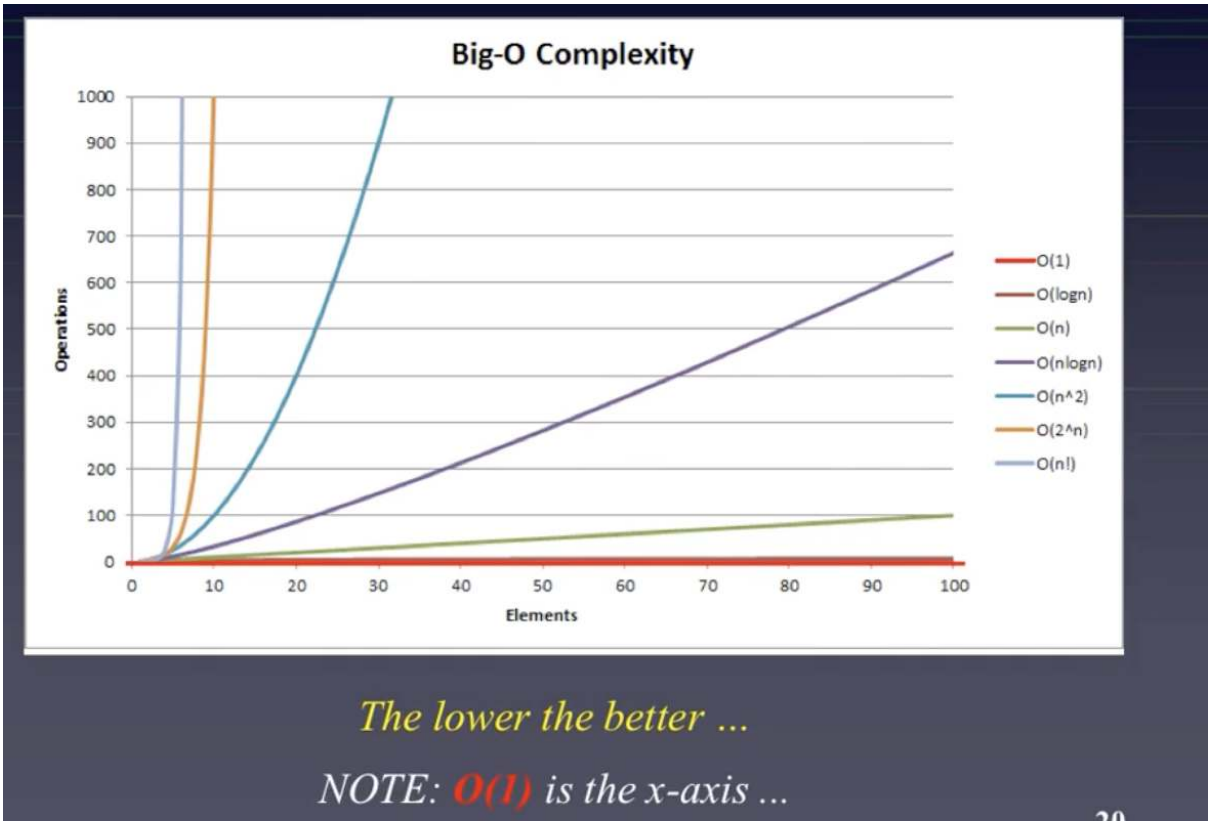
- I want to read block #6?
  ➔ read 1, then 2, then 3, ... then block 6
  i.e. need 6 reads to get to block 6!

- Indexed O(log n):
  - this is shortest path in a tree
    1. Read the "index block"
       1. If required block is in the data index then you have the block# ➔ 2 reads needed
       2. If required block is in 2nd level index, then ➔ read the 2nd index block – if in the index then you have the block# ➔: 3 reads needed



13

Binary search



The lower the better ...

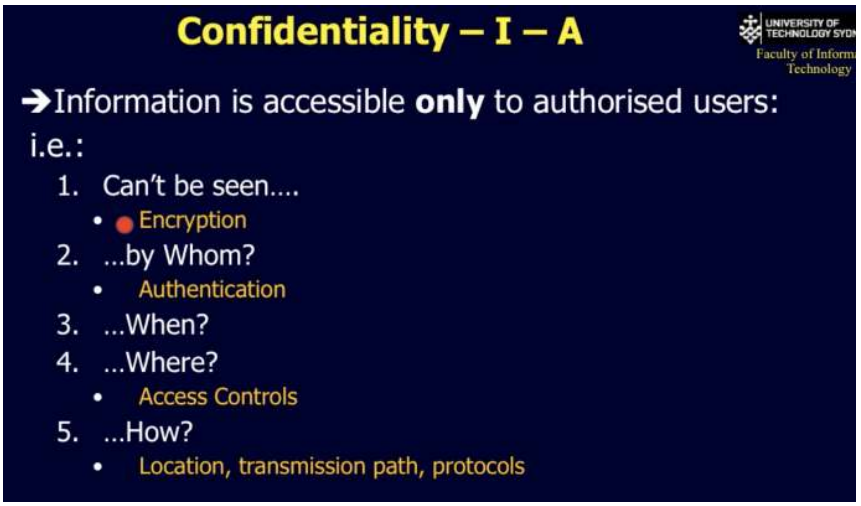
NOTE: O(1) is the x-axis ...

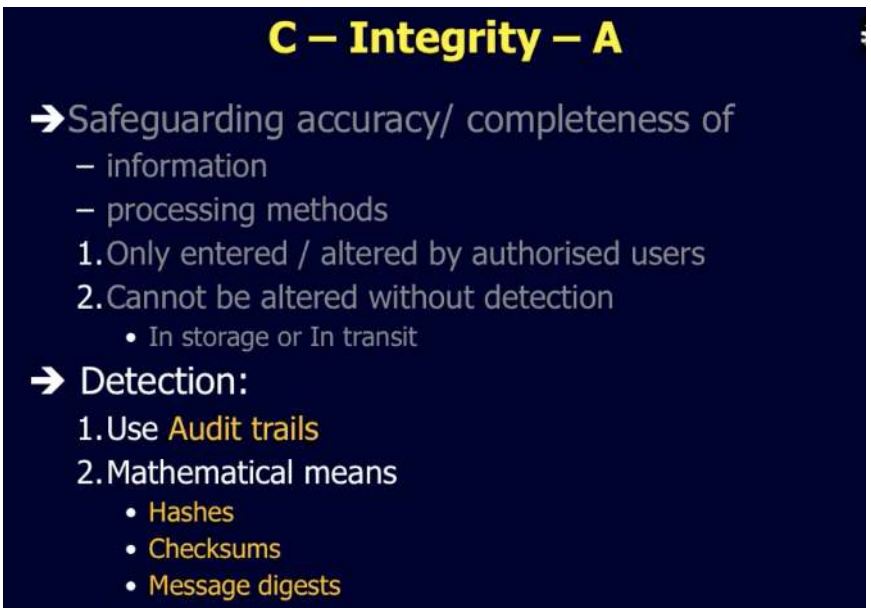
Complexity Theory

- Try this with Laptop sized filesystem
  - 400,000 files, **178 Gb**
- Contiguous: O(1) "about 1" (not likely but..)
  - Works on SSD quite well... *Except for fragmentation!*
- Chained/Linked: O(n) "roughly n blocks"
  - 1st block – need 1 read.
  - BUT 98 millionth block – need **98 million** reads!
- Indexed/Inode: O(logk(n)) blocks
  - $Log_{10}(178x10^9) \approx$ **11** reads needed!

More OS, and Introduction to the Web - Week 4

The web and security





C – I – Availability

- Ensuring authorised users have access to information/processing when required

i.e.:

1. Systems survive failures
   - Have hot/cold standby mechanisms
2. Systems resist attacks
   - Resistant to Denial of Service (DoS) attacks

- Users can access from authorised locations

Good vs Evil

Security Service
- A security service makes use of **one or more** security mechanisms

Security Attack
- Any action that compromises the security of information.

- **Security Mechanism:**
  - A mechanism that is designed to detect, prevent, or recover from a security attack

Types of attacks

- Interruption: Availability attack
- Interception: Confidentiality attack
- Modification: Integrity attack
- Fabrication: Authenticity attack

Typical Security Services

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Information Technology

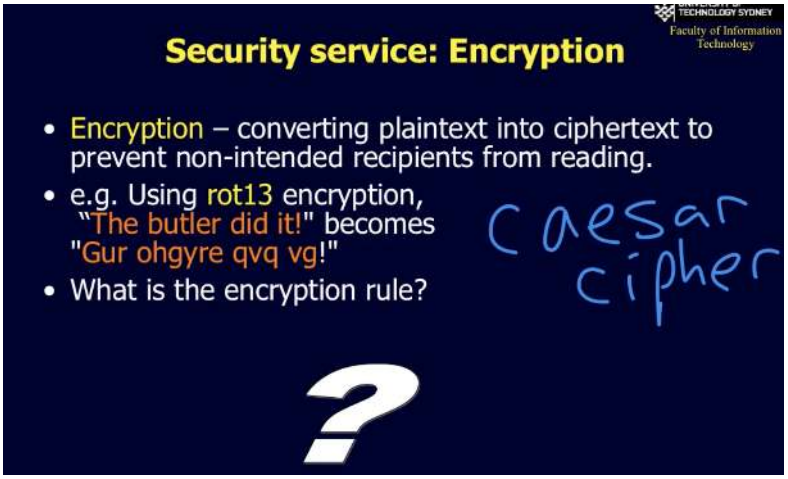
- Confidentiality – privacy - encryption
- Authentication - who created or sent it
- Integrity - has not been altered
- Non-repudiation - the order is final
- Access control - prevent misuse of resources
- Availability - permanence, non-erasure

Paypal

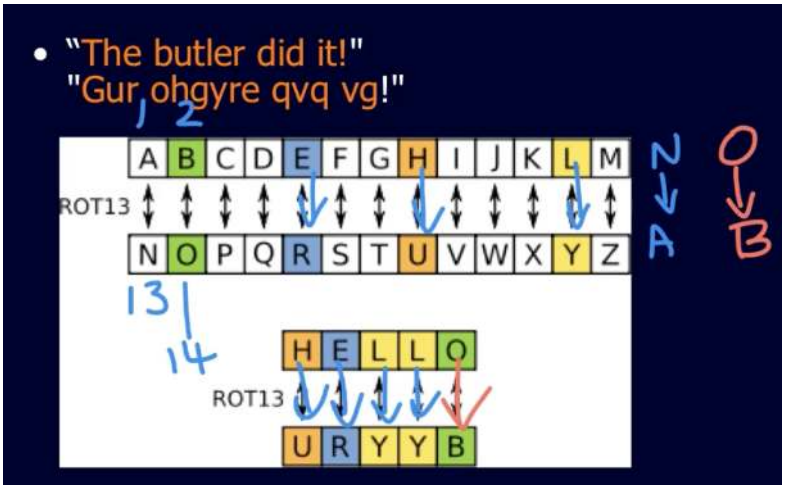
Levels Of Access?

$ $$$

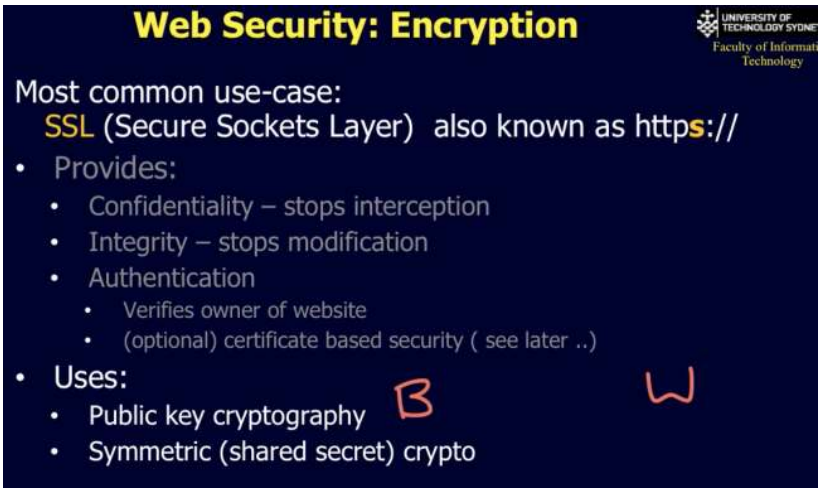
Security Services



Shifts the character by 13

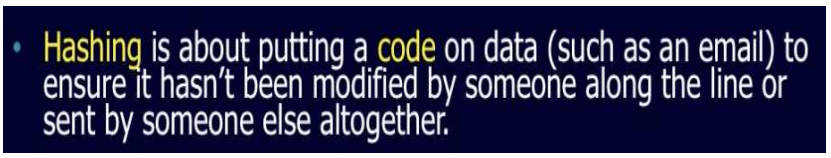


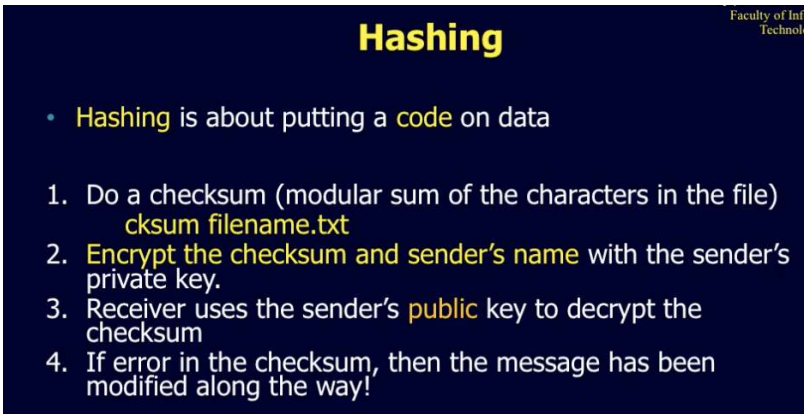
Problem can solved through incorporating Public Key Cryptography



Browser has half a key and sever has the other part of the key

Stopping modification through hashing





Add up all the characters and the encrypt the number, if the number isnt same upon decryption, something has been modification.

Web Security: hashing

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Information Technology

Typically used in

- Secure email
  - some email clients can verify that your email was not tampered → need personal digital certificate
- Electronic documents
  - Adobe PDF allows you to digitally sign documents
- Validate software
  - when installing software on windows, the installer must be signed by Microsoft

Authentication and access control

Authentication

- Something you **know**
  - Password
  - PIN
- Something you **have**
  - Key
  - Token
  - Certificate
- Something you **are**
  - Fingerprint, palm print
  - Retinal scan
  - Face recognition
  - Voice recognition

Where do we Authenticate?

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Information Technology

- At the perimeter? WWW door
- At the operating system?
- At the Application?
- Are there multiple processes for the users? PW / PIN card
  - Eg: staff who are students?? staff / 7 sid
- Multiple user lists to coordinate and manage!!!

Basic authentication

Access control

UNIX security: Access Control

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Information Technology

- Usually 3 levels of security:
  - **u**ser → owner of the file
  - **group** → other users in the owner's group
  - **others** → the public
- Each file and directory has 3 sets of permissions
  - **r**ead → can read the file
  - **w**rite → can update the file
  - e**x**ecute → can execute (if a file) or traverse (if a directory)
- Permissions appear like:
- `drwxrwxrwx  53 chw  staff 450 Apr 1 00:01 public_html/`

Security Mechanism: Audit Log

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Information Technology

- Audit trails / logs are essential
- Needed to
  - Measure effectiveness
  - Do forensics
  - Create alerts
- Also subject to security needs!

violation

CW X pay

Availability and risks

C – I – Availability

- Ensuring authorised users **have access** to information/processing **when required**

1. Systems survive failures?
   - Have hot/cold standby mechanisms
   - Backups
2. Systems resist attacks?
   - Firewalls
   - CDN vs Distributed Denial of Service (DDoS)
   - Anti-malware

Survive Failures

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Informatic Technology

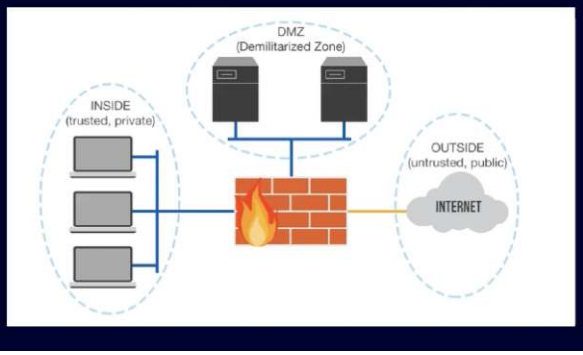
- Good design & good operations!
  - Have hot/cold standby mechanisms
  - redundancy
    - Have more than 1 server ...
    - Have more than 1 data centre ...
    - Have more than 1 network connection ... etc etc
  - Disaster recovery –
    - "hot" standby – online system kept in sync.
    - Also → can assist when super busy e.g.: discount sales events! Can swap in minutes...
    - "cold" standby – not-online but can be started up quickly (hours?)

Backups - On site, off site.

Resist Attacks - Firewall

UNIVERSITY OF TECHNOLOGY
Faculty of In Techno

- Good practice – stop threats crashing your system.
  - **Firewalls**
    - At the perimeter (network)
    - Sometimes consider DMZ for external facing servers



- Good practice – stop outages crashing your system.
  USE REDUNDANCY!
  - Use multiple network connections from different ISP!
  - Protect against Distributed Denial of Service (DDoS) via network infrastructure
    - Specialised routing/switch hardware
    - Content Delivery Networks

- Good practice – stop outages crashing your system.
  VACCINATE!!!
  - Install Anti-malware software
  - Monitor via security scanning systems
  - Training staff/users
  - Protect against social-engineering!

Security & Risk Assessment

- Security should match level of Risk Assessment!
- Internal or external threat sources?
- Vandalism?
  - Can be malicious
  - Can be politically motivated
- Industrial espionage?
- Theft?