Definitions

IEEE Floating Point	±0: e, m = 0 ±∞: e = max, m = 0 NaN: e = max, m ≠ 0 Denorm: e = 0, m ≠ 0

Multiprogramming

Scheduling	First come, first served Shortest job first (optimal) Shortest remaining time first (new process w/ low burst: preempt) Round robin (with quantum)
e Average	$\tau_{n+1} = at_n + (1-a)\tau_n$
Priority	Static: e.g. use priority as next predicted burst time Dynamic: aging (prevent starving) or computed: penalise time use

Memory

Memory	Static: fixed size partitions Dynamic: partition at runtime Paged: physical frame to logical page mapping
Compaction	Run time relocation Do it when you move off swap
Replacement	FIFO: Belady's anomaly etc LRU: timestamp/page stack NRU: reference/dirty bit or second chance FIFO (clock) Reference counting Page buffering (pool of victims) App-specific hooks Locality of reference "Working set" <=> thrashing
Segments	Local/global page tables External fragmentation Software segments (page array, OS keeps priv. consistent) Paged segments (per-segment page tables, not portable)

I/O

Access Modes	Polled vs interrupt driven Blocking/nonblocking/asynch
Buffering	Maintains copy semantics Single/double/circular Sized according to device type
Other issues	Caching, scheduling (queue/fairness), device reservation, error handling
File Issues	Directory service (name -> id) Storage service (id -> data) The DS must be implemented on top of the SS (obviously)
Directories	In a directed <u>acyclic</u> graph Directories stored as files open/create (SFID -> UFID)
FS Issues	Access control Existence control (GC) Concurrency control: locks. Mandatory/advisory, shared/exclusive

Protection

Goals	Prevent information disclosure/modification Denial of service Isolation (debug/error control)
Mechanisms	User/supervisor modes Memory management control File control (ACLs etc) Physical restrictions Passwords/encryption Stupidity/legislation
Principles	Least privilege Default deny Current authority (caching..) Psychologically acceptable High circumvention cost
Authentication	Passwords/biometrics/cards Mutual suspicion
Access Matrix	Keyed on object -> ACL Keyed on subject -> capability
Capabilities	Address space storage -> hardware access Machine instructions to modify Software caps checked by encryption/timeouts Hybrid: key ACL (stored at resource) on capability

Unix File System

Inodes	Type/mode/user|groupid/size/nlinks Direct x12/single|doub|trip indirect
Directory	Files with inodes holding list of SFID Can have at most 1 hard link
Disk	Boot|super|inode table|data blocks Superblock: nfree, free link lists etc Can "mount" into name service
Files	Descriptor table: process specific -> system wide -> device inode table UGO bits + setg|uid. Directories use X = cwd, SG = group "sticky" Consistency issues (on crash)

Unix Processes

Principles	Heavyweight (own page table, are the unit of scheduling) Shared kernel space -> no c-switch Zombie state (for parents benefit)
Boot	Kernel -> init -> tty -> login -> sh
IPC	Pipes (later named pipes): consist of finite circular queue Signals which process can catch
I/O	Buffer cache w/ sync every 30 seconds Aggressive metadata writeback
Scheduling	Lower priorities superuser only Penalises CPU usage over ≈ 5s

Windows Architecture

Structure	Super: HAL, kernel, executive User: environment/protection subs. HAL: interrupt/DMA/SMP etc Kernel: no pre-emption, schedules, handles interrupts, processor sync.
Processes	Processes own resources Threads are dispatch units, lightweight and share resources Parent/child not mandatory
Scheduling	Boost on return from IO/fg thread Priority decays over time to base Also get static priority (“real time”)
Objects	Object manager checks ACLs/creates objects|handles Implies uniform security model enforced by Security Ref. Manager Name, directory, security descriptor, type|info, ref count Live in a namespace with recursive name parsing responsibility
VMM	Can share memory in section object = segment (based|non)
IPC	Channels (copy, zero-copy, quick)
I/O	Asynchronous: IRP holds parameters, results etc Stackable drivers handle IRPs “Virtual block” cache w/ prefetch Unified cache works on VAS “lines” (VMM does cache I/O) User control (temp/write through)
Subsystems	Layered over NT native API DOS, OS/2, POSIX, WoW

Windows File Systems

FAT16	Linked list of clusters: max 2Gb Variable cluster size
FAT32	Wider FAT16: 8Gb @ 4k cluster Root directory anywhere on disk Can use backup FAT (fault tolerant) VFAT: long names on top of this
NTFS	File records held in MFT (itself a file) indexed on file ref (64 bits) Based on a volume, not partition Files are attribute/value pairs Special: LogFile, Bitmap, BadClus Transactions for consistency Volumes can be RAID sets, supporting bad cluster remapping Security descriptor in MFT Compression and sparsity Symmetric encryption w/ RSA key on that key, admin can get key..