NSCT Operating Systems Cheat Sheet

By the NSCT Exam Editorial Team · Updated 2026-11-24 · Facts checked against official sources on 2026-11-24

Operating Systems carries 5% of the NSCT according to HEC's Areas of Competencies, and it was the weakest area in 2026 at 33.8% average accuracy, per the HEC NSCT report and APP. Print this page and keep it next to your practice: most OS marks come from a handful of fixed formulas, and terms follow Silberschatz.

Key definitions

Term	One line
Process	Program in execution (the program itself is passive)
PCB	State, PID, program counter, registers, scheduling, memory and I/O info
Thread	Own ID, PC, registers and stack; shares code, data, heap and files
Trap	Software-generated interrupt (error or system call)
Mode bit	Kernel = 0, user = 1
Critical-section needs	Mutual exclusion, progress, bounded waiting
Semaphore	Integer with `wait()` (P, decrement) and `signal()` (V, increment); no owner
Deadlock	Set of processes each waiting for a resource another holds
Thrashing	Fewer frames than the working set; more paging than computing
Belady's anomaly	More frames, more faults; affects FIFO, not LRU or Optimal
Inode	Attributes and block pointers, not the file name
DMA	Device moves a whole block; one interrupt per block

Formulas

Formula	Variables and notes
TAT = completion − arrival	Turnaround time
WT = TAT − burst	Waiting time in the ready queue
Response = first run − arrival	Time to first service
CPU utilisation = $1 - p^n$	p = fraction of time waiting on I/O, n = processes in memory
n `fork()` calls in sequence = 2^n processes	$2^n - 1$ children; `fork() && fork()` gives 3
Amdahl speed-up ≤ 1 / (S + (1 − S) / N)	S = serial fraction; ceiling 1 / S
Need = Max − Allocation	Banker's; grant only if Request ≤ Need and ≤ Available
No deadlock if total ≥ n(k − 1) + 1	n processes, each needing at most k units
Physical address = frame × page size + offset	Offset bits = $\log_2$(page size)
Page-table entries = 2^(logical bits) ÷ page size	Size = entries × entry size
EAT (TLB) = h(TLB + mem) + (1 − h)(TLB + 2·mem)	Silberschatz 10th edition ignores TLB time
EAT (faults) = (1 − p) × mem + p × fault time	Convert ms to ns first
Access time = seek + latency + transfer	Average latency = 0.5 × 60 / RPM seconds
Inode max = (direct + k + k^2 + k^3) × B	k = B ÷ pointer size
Bitmap size = blocks ÷ 8 bytes	One bit per block
RAID 5 = (n − 1) × c; RAID 6 = (n − 2) × c; RAID 10 = n × c / 2	n disks of capacity c

Worked numbers to check yourself

Question	Answer
TLB 20 ns, memory 100 ns, h = 0.9	0.9 × 120 + 0.1 × 220 = 130 ns (110 ns if TLB time ignored)
Memory 100 ns, fault 10 ms, p = 0.0001	0.9999 × 100 + 0.0001 × 10,000,000 ≈ 1,100 ns
32-bit logical address, 4 KB pages, 4-byte entries	2^{20} entries, 4 MB page table, 12 offset bits
p = 0.8, n = 4	Utilisation = 1 − 0.4096 ≈ 59%
10 ms slice, 1 ms switch	1/11 of CPU time is overhead
3 processes, each needing up to 3 units	3 × 2 + 1 = 7 units prevent deadlock
7,200 RPM	Average latency ≈ 4.17 ms
4 KB blocks, 4-byte pointers, 12 direct	k = 1024; max ≈ 4 TB
4 disks × 2 TB	RAID 5 = 6 TB; RAID 6 = 4 TB

Scheduling at a glance

Algorithm	Pre-emptive?	Remember
FCFS	No	Convoy effect; no starvation
SJF	No	Minimum average waiting time; can starve
SRTF	Yes	Pre-emptive SJF; can starve
Priority	Either	Starves; fix with aging; check whether low number = high priority
Round robin	Yes	Huge quantum = FCFS; better response, usually worse TAT than SJF

Disk scheduling (Silberschatz queue)

Queue 98, 183, 37, 122, 14, 124, 65, 67; head at 53; cylinders 0–199.

Algorithm	Rule	Total head movement
FCFS	Arrival order	640
SSTF	Nearest request next (can starve)	236
SCAN, moving down	Sweep to cylinder 0, then reverse	236
LOOK, moving down	Reverse at the last request (14)	208
C-SCAN, moving up	Sweep to 199, jump to 0, continue up	382 counting the return, 183 without
C-LOOK, moving up	Jump back to the lowest request (14)	322 counting the jump

X vs Y quick distinctions

Pair	Difference
Process vs thread	Own address space vs shared address space, own stack
Monolithic vs microkernel	Services in kernel, fast vs user-space servers, safer, slower
Pre-emptive vs non-pre-emptive	CPU can be taken away vs runs until the burst ends or blocks
Mutex vs semaphore	Has an owner, locked/unlocked vs no owner, integer count
Deadlock vs starvation	Group stuck in a cycle vs one process never served
Prevention vs avoidance	Deny a Coffman condition vs check each request (Banker's)
Paging vs segmentation	Fixed size, internal fragmentation vs variable, external
SCAN vs LOOK	Reverses at the end cylinder vs at the last request
Hard vs symbolic link	Same inode, same file system vs stored path, can dangle
Access list vs capability list	Matrix column per object vs row per domain
Polling vs interrupts vs DMA	CPU busy-waits vs interrupt per word vs interrupt per block
Nonblocking vs asynchronous I/O	Returns what is ready now vs completes later and notifies

Mnemonic for the Coffman conditions: mutual exclusion, hold and wait, no pre-emption, circular wait. All four are needed; break one to prevent deadlock.

Classic traps

1. Waiting → running and ready → waiting never happen; finished I/O makes a process ready.
2. A root program still runs in user mode; kernel mode is a CPU state.
3. Loadable modules do not make Linux a microkernel.
4. FCFS and round robin cannot starve a process.
5. 6 user threads on 3 kernel threads run at most 3 at once.
6. In producer-consumer, `wait(mutex)` before `wait(empty)` deadlocks on a full buffer.
7. A RAG cycle means deadlock only with single-instance resources; unsafe is not deadlocked.
8. Check a Banker's request against Need before Available.
9. 10 ms is 10,000,000 ns, not 10,000.
10. SCAN goes to the end cylinder; stopping at the last request is LOOK.
11. Reaching block i under linked allocation takes i reads.
12. Check whether the question counts the C-SCAN return seek.

Keep going

- Full explanations: Operating Systems revision notes
- All 12 topics and quizzes: Operating Systems MCQs hub
- Deep pages: CPU scheduling numericals, page replacement, Banker's algorithm, paging and address translation, disk scheduling, semaphores
- Timed practice: subject-wise mock test

Next step

Rework every worked number with this page covered, then take a 20-question Operating Systems set in the subject-wise mock test. For more calculation drills across subjects, use numericals practice, and finish with a full NSCT mock test.