IAE ACADEMY INSTITUTE FOR ENGINEERS

GATE CS

IAE Academy free study material · Updated 13 October 2026

Operating Systems Formula Sheet

INSIDE

- 47 formulas in 7 topics
- When to use each one
- Taken from the free topic notes on iaeacademy.in

Online, one page per topic: **iaeacademy.in/study/operating-systems/formula-sheet**

FREE COUNSELLING

Preparing for GATE CS? Talk to IAE Academy.

Or call **+91 84204 36524**

Every day, 10:30 am to 7:30 pm

Chat on WhatsApp

Free. Not for sale. Pass it on.

Processes, threads and system calls

TOTAL PROCESSES

$$N_{\text{total}} = 1 + F$$

F is the number of fork() calls that actually execute, summed over all processes; assumes every fork succeeds.

N UNCONDITIONAL FORKS IN A ROW

$$N_{\text{total}} = 2^n$$

n fork() calls one after another with no condition and no exit; the same holds for a loop that runs n times with a fork in it and no break. New (child) processes created = 2^n - 1.

LINES PRINTED BY A PRINTF AFTER A FORK INSIDE AN N-ITERATION LOOP

$$L = 2^1 + 2^2 + \cdots + 2^n = 2^{n+1} - 2$$

For a loop body of fork(); followed by printf("...\n"); with every process continuing the loop and each printf appearing once per process that executes it: iteration k (counting from 1) is executed by 2^k processes.

CPU EFFICIENCY WITH CONTEXT-SWITCH OVERHEAD

$$\eta = \frac{q}{q + s}$$

q is the length of the useful time slice before each switch and s is the context-switch time, in the same unit; assumes every slice is used in full. If each process instead runs only for T < q before blocking, use T in place of q.

Inter-process communication, concurrency and synchronisation

INTERLEAVINGS OF TWO INSTRUCTION SEQUENCES

$$\binom{m+n}{m}$$

Number of distinct interleavings of two independent sequences of m and n instructions, each sequence keeping its own order. Different interleavings may still give the same output.

CONDITION FOR TWO STATEMENTS TO RUN CONCURRENTLY

$$R(S_1) \cap W(S_2) = \emptyset, \quad W(S_1) \cap R(S_2) = \emptyset, \quad W(S_1) \cap W(S_2) = \emptyset$$

R(S) is the set of variables statement S reads and W(S) the set it writes. If all three intersections are empty, S1 and S2 give the same result whichever runs first or if they run in parallel.

SEMAPHORE VALUE INVARIANT

$$S = S_0 - n_w + n_s$$

For the blocking form above: S_0 is the initial value, n_w the number of wait operations executed (including blocked ones) and n_s the number of signal operations executed. If S < 0 then -S processes are blocked. (In the busy-waiting form, n_w counts completed waits only.) Some questions use a form in which wait blocks without decrementing when S = 0, and signal wakes a blocked process instead of incrementing. There the final value is max(0, S_0 + n_s - n_w) and the number blocked is max(0, n_w - n_s - S_0).

Deadlock: conditions, prevention, avoidance and detection

NEED MATRIX

$$\text{Need}[i][j] = \text{Max}[i][j] - \text{Allocation}[i][j]$$

Remaining claim of process P_i for resource type R_j.

AVAILABLE VECTOR

$$\text{Available}[j] = \text{Total}[j] - \sum_{i=0}^{n-1} \text{Allocation}[i][j]$$

Free instances of R_j; Total is the number of instances of R_j in the system.

MINIMUM RESOURCES THAT GUARANTEE NO DEADLOCK

$$R_{\min} = n(k-1) + 1$$

Single resource type, n processes, each needing at most k units (and possibly asking for all k), units requested one at a time and held while waiting. Deadlock is impossible iff $R \geq n(k-1)+1$; with $R \leq n(k-1)$ a deadlock can occur.

LARGEST NUMBER OF PROCESSES WITH NO POSSIBLE DEADLOCK

$$n_{\max} = \left\lfloor \frac{R-1}{k-1} \right\rfloor$$

Same assumptions, $k \geq 2$. Follows from $n(k-1)+1 \leq R$.

CPU scheduling

TURNAROUND TIME

$$\text{TAT} = \text{CT} - \text{AT}$$

Holds for every scheduling algorithm.

WAITING TIME

$$\text{WT} = \text{TAT} - \text{BT}$$

Valid when the process has a single CPU burst and does no I/O. With I/O bursts, WT = TAT − (total CPU time) − (total I/O time).

RESPONSE TIME

$$\text{RT} = (\text{time of first dispatch}) - \text{AT}$$

In a non-preemptive algorithm, RT equals WT when each process has a single CPU burst and does no I/O.

UTILISATION AND THROUGHPUT

$$\text{CPU utilisation} = \frac{\text{time CPU is busy with processes}}{\text{total elapsed time}}, \qquad \text{Throughput} = \frac{n}{\text{total elapsed time}}$$

Total elapsed time runs from time 0 to the last completion unless the question defines it otherwise. Idle gaps and context-switch time count as not busy.

HRRN RESPONSE RATIO

$$\text{Response ratio} = \frac{W + S}{S}$$

W = time the process has waited so far in the ready queue, S = its burst time. Recompute it at every instant the CPU becomes free.

EXPONENTIAL AVERAGING

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\, \tau_n, \qquad 0 \leq \alpha \leq 1$$

t_n = actual length of the nth CPU burst, tau_n = its predicted value, tau_1 = the given initial guess.

RR TURN BOUND

$$\text{Gap between two turns of a process} \le (n-1)\,q + n\,s \quad \Rightarrow \quad q \le \frac{T - n\,s}{n-1}$$

For n processes that all keep using full quanta, with a context switch of s ms between any two processes. The second form is the largest quantum for which every process gets the CPU again within T ms. With s = 0, the k-th process in a queue formed at time 0 first gets the CPU within (k − 1)q.

RR EFFICIENCY WITH SWITCH TIME

$$\text{CPU efficiency} = \frac{q}{q+s}$$

Valid when every process uses its full quantum q and each dispatch costs a context switch of s ms.

CPU UTILISATION WITH OVERHEADS

$$\text{Utilisation} = \frac{\sum \text{CPU burst times}}{\sum \text{CPU burst times} + \text{switch time} + \text{idle time}}$$

Use when context-switch time and idle gaps are given or can be read from the Gantt chart. The denominator is the total elapsed time.

Memory management: allocation, paging and segmentation

ADDRESS SPLIT

$$d = \log_2(\text{page size}), \qquad \text{number of pages} = \frac{2^n}{2^d} = 2^{\,n-d}$$

n is the logical address width in bits; the page offset has d bits and the page number n-d bits.

FRAMES

$$\text{frames} = \frac{\text{physical memory size}}{\text{page size}}, \qquad \text{frame-number bits} = \log_2(\text{frames})$$

Physical address = frame number bits followed by d offset bits.

SINGLE-LEVEL PAGE TABLE SIZE

$$\text{page table size} = (\text{number of entries}) \times e$$

e is the size of one page-table entry in bytes; the number of entries equals the number of pages in the logical address space.

MINIMUM PAGE-TABLE ENTRY WIDTH

$$\text{entry bits} \ge \text{frame-number bits} + \text{extra bits}$$

Extra bits are those the question lists (valid, dirty, protection and so on). Round up to whole bytes; round to a power of two only if the question says so.

NUMBER OF LEVELS

$$\text{entries per table} = \frac{\text{page size}}{e}, \qquad k = \left\lceil \frac{n-d}{\log_2(\text{page size}/e)} \right\rceil$$

k is the minimum number of levels when each table occupies one page; n-d is the number of page-number bits. If n-d is not a multiple of the bits per level, the convention used here gives the leftover bits to the outermost table, which then occupies less than a page.

INVERTED PAGE TABLE SIZE

$$\text{inverted table size} = \frac{\text{physical memory}}{\text{page size}} \times e$$

One entry per frame; e is the entry size in bytes (an entry holds the process id and page number).

EFFECTIVE ACCESS TIME WITH A TLB

$$\mathrm{EMAT} = h\,(t + m) + (1 - h)\,\big(t + (k + 1)\,m\big)$$

t = TLB lookup time, m = main-memory access time (same unit as t), k = number of page-table levels walked on a miss. The TLB is searched first on every reference, and a miss then costs k extra memory accesses. Assumes no page faults and page-table entries not cached elsewhere. If a question says the TLB lookup overlaps or is ignored, set t = 0.

ACCESS TIME WITHOUT A TLB

$$\mathrm{EMAT}_{\text{no TLB}} = (k + 1)\,m$$

k page-table accesses plus one data access.

SEGMENT TRANSLATION

$$\text{physical address} = \text{base}[s] + o \quad \text{if } o < \text{limit}[s]$$

Offsets run from 0 to limit - 1; an offset equal to the limit is out of range.

Virtual memory: demand paging and page replacement

EFFECTIVE ACCESS TIME

$$\mathrm{EAT} = (1 - p)\,m + p\,S$$

Weighted average of a hit-path access (time m) and a faulting access (total time S).

EFFECTIVE ACCESS TIME WHEN THE FAULT TIME IS EXTRA

$$\mathrm{EAT} = (1 - p)\,m + p\,(m + S) = m + p\,S$$

Use this form when the question gives S as a delay added on top of the normal memory access, so a faulting access costs m + S.

AVERAGE FAULT SERVICE TIME WITH DIRTY VICTIMS

$$S = (1 - d)\,S_{\text{clean}} + d\,S_{\text{dirty}}$$

d is the fraction of faults whose victim is dirty; S_clean and S_dirty are the service times for a clean and a dirty victim.

LARGEST TOLERABLE FAULT RATE

$$p \le \frac{\varepsilon\,m}{S - m}$$

Keeps EAT at most (1+ε)·m, i.e. a slowdown of at most 100ε percent, with S the total time of a faulting access. Follows from (1-p)m + pS ≤ (1+ε)m.

PROPORTIONAL ALLOCATION

$$a_i = \frac{s_i}{\sum_j s_j} \times m$$

s_i is the size (in pages) of process i and m the number of available frames. If a share is not whole, use the rounding the question states (for example, round down and keep the leftover frames free), keeping each share at or above the minimum.

TOTAL FRAME DEMAND

$$D = \sum_i \mathrm{WSS}_i$$

If D is greater than the number of available frames m, thrashing is likely; the OS should suspend a process (freeing its frames). If D is at most m, the OS may start another process.

File systems and disk scheduling

LINKED ALLOCATION

$$\text{usable bytes per block (linked)} = B - p, \qquad \text{blocks for a file of } S \text{ bytes} = \left\lceil \frac{S}{B - p} \right\rceil$$

Applies when each data block stores its own next-block pointer of p bytes.

WIDTH OF A BLOCK ADDRESS

$$\text{bits per block address} \geq \lceil \log_2 N_{\text{blocks}} \rceil$$

A pointer or FAT entry must be able to name every one of the N_{blocks} blocks; questions usually round this up to whole bytes.

FAT SIZE

$$\text{FAT size} = N_{\text{blocks}} \times (\text{entry size in bytes})$$

One entry per disk block; the entry must be wide enough to hold any block number.

POINTERS PER BLOCK

$$P = \frac{B}{p}$$

Number of block addresses that fit in one index (indirect) block.

LARGEST FILE WITH AN I-NODE

$$\text{Max file size} = \left(D + P + P^2 + P^3\right) \times B$$

With direct, single, double and triple indirect pointers. Drop the terms for levels the i-node does not have. This is the limit imposed by the pointer scheme; a pointer width or file-size field can impose a smaller limit.

BIT VECTOR SIZE

$$\text{Bit vector size (bytes)} = \frac{N_{\text{blocks}}}{8} = \frac{\text{disk size}}{8 \times B}$$

One bit per block of size B bytes.

DISK CAPACITY

$$\text{Capacity} = \text{surfaces} \times \frac{\text{tracks}}{\text{surface}} \times \frac{\text{sectors}}{\text{track}} \times \frac{\text{bytes}}{\text{sector}}$$

Assumes the same number of sectors on every track.

LINEAR SECTOR NUMBER FROM (CYLINDER, HEAD, SECTOR)

$$\text{sector number} = (c \times H + h) \times S + s$$

H heads (surfaces), S sectors per track, and c, h, s all numbered from 0, with sectors numbered across a whole cylinder before moving to the next cylinder. If sectors are numbered from 1, use $s - 1$ in place of s.

DISK ACCESS TIME

$$T_{\text{access}} = T_{\text{seek}} + T_{\text{rotational latency}} + T_{\text{transfer}}$$

Add controller or queueing overhead only if the question gives it.

ROTATION TIME AND AVERAGE LATENCY

$$T_{\text{rot}} = \frac{60}{\text{RPM}}\ \text{s}, \qquad \text{average rotational latency} = \frac{T_{\text{rot}}}{2}$$

On average the wanted sector is half a turn away. If the question says worst case, use a full T_{rot}.

TRANSFER TIME

$$T_{\text{transfer}} = \frac{\text{bytes to transfer}}{\text{bytes per track}} \times T_{\text{rot}}$$

The head reads one full track per rotation, so time is the fraction of a track passed under the head times the rotation time.

DATA TRANSFER RATE

$$\text{data transfer rate} = \text{bytes per track} \times \frac{\text{RPM}}{60} \ \text{bytes/s}$$

One track passes under the head per rotation.

FROM IAE ACADEMY

Preparing for GATE CS?

IAE Academy teaches GATE CS in Ballygunge, Kolkata, and online. Counselling is free: ask us about batches, fees or your preparation.

FREE COUNSELLING

Chat on WhatsApp

Call +91 84204 36524

Open every day, 10:30 am to 7:30 pm.

Scan to chat on WhatsApp

THE GATE CS COURSE

Classroom batches in Ballygunge and online batches. Fees and details are on the course page.

Scan: course page

See the course

FREE ON IAEACADEMY.IN

Notes and solved previous-year questions for Operating Systems, topic by topic.

Scan: the notes

Open the notes

Found this useful? Send it to a friend preparing for GATE.

Send to a friend on WhatsApp

IAE Academy free study material. The formulas come from the free Operating Systems notes on iaeacademy.in. Prepared and published by IAE Academy, Kolkata. Free to read and free to share, unaltered and with attribution.

IAE ACADEMY INSTITUTE FOR ENGINEERS

GATE coaching in Kolkata since 2012 · EC, EE, IN, CS and IT, DA, BT and CE · in the classroom and online

227A Rash Behari Ave, Ballygunge, Kolkata 700019

+91 84204 36524 · iaeacademy.in · Updated 13 October 2026

Get directions