

CSE101 · Week 4

Operating Systems and Processes

Prof. Dr. Mehmet Göktürk
Gebze Technical University · Fall 2026

A computer was a shared facility



IBM 704 at NACA Langley, 1957. Who decides which job gets the machine next?

Many users, one computing service



Terminal classroom at UC Santa Barbara, 1966. Users interact through terminals.

Sharing the machine

Week 2: the CPU executes instructions and communicates with devices.

Now: several programs need CPU time, memory and I/O.

The operating system manages access to these resources.

Week 3 traced one program. Now we explain how several processes share the machine.

A few cores, hundreds of processes

A typical PC has one CPU package containing several processing cores.

Desktop example: Intel Core Ultra 5 225 has 10 cores in one processor.

Hundreds of processes may exist.
The count depends on apps and services.
Most are waiting, not executing.



Historical board with two CPU packages.
Each package here is a separate processor.

Many processes share a small number of execution resources.

The mechanic-shop analogy



Mechanic = CPU/core. Each car's repair job = a process in progress.

One mechanic, several repair jobs

The mechanic works on one car at a time.

A repair can pause while a spare part is being delivered.

The mechanic can work on another car while waiting.

Several jobs make progress over time: concurrency.

Two mechanics: parallel work



Two mechanics can work simultaneously: two cores or two processors.

Concurrency and parallelism

Concurrency: multiple tasks make progress during overlapping periods.

One core can alternate between tasks; they need not run simultaneously.

Parallelism: separate execution resources work at the same time.

Two mechanics illustrate parallel execution, not CPU packaging.

Priority: which job goes first?



A police vehicle illustrates a possible priority rule, not a universal rule.

Choosing the next job



The coordinator chooses work for the mechanic, like a CPU scheduler.

Short jobs



A short repair may finish quickly, but the next job still needs a selection rule.

Long jobs



A long repair may use many periods of work before it is complete.

Waiting for service



Waiting for a mechanic is different from waiting for a missing spare part.

Programs and processes

A program is a stored sequence of instructions.

A process is an executing instance of a program, with its own state.

Two runs of the same program can have different inputs and progress.

In the analogy, the repair procedure is the program; each repair is a process.

Three process states

State	Meaning	Shop analogy
Running	Using the CPU	Mechanic working on car
Ready	Can run; awaiting CPU	Car ready for service
Blocked	Waiting for an event	Waiting for spare part

When the part arrives, the job becomes ready; it does not automatically run.

The scheduler selects a ready process



Scheduling is a choice: which eligible job should receive CPU time?

The dispatcher gives it the CPU



Dispatching carries out the switch so the selected process can resume.

Saving a process's place

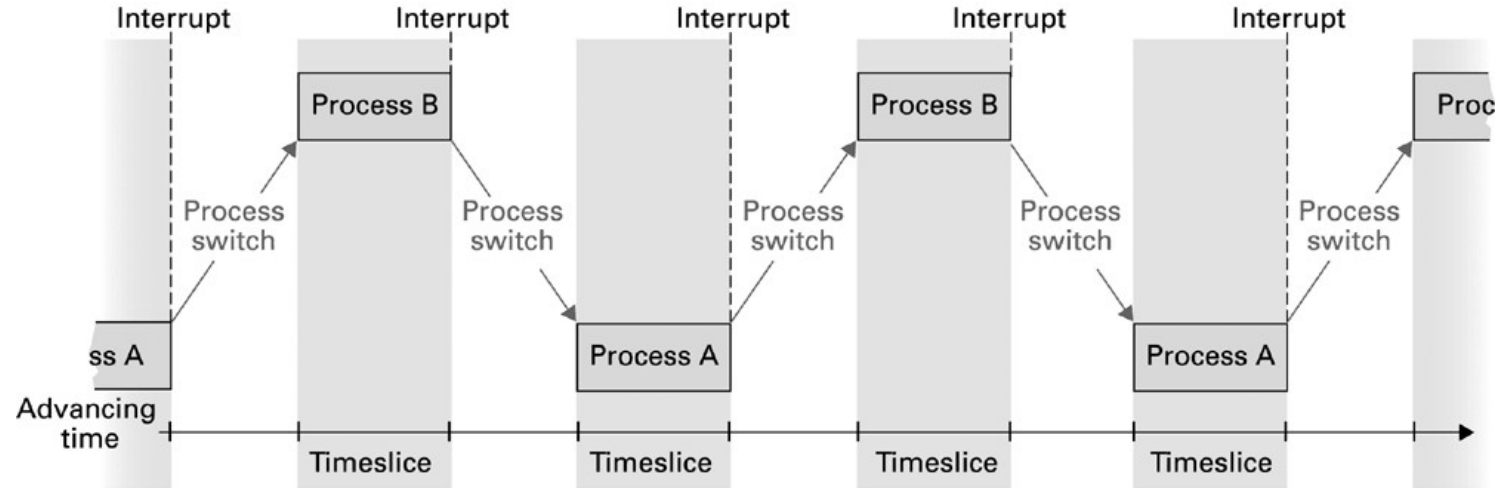
Record the next instruction address: the program counter.

Save register values needed when execution resumes.

Keep track of the process's memory and other resources.

A context switch saves one execution context and restores another.

Sharing CPU time



Timer interrupts let the OS regain control and consider another ready process.

Interrupts and waiting

A timer interrupt can end a process's turn on the CPU.

A device interrupt can report that an I/O operation has completed.

Completion may make a blocked process ready again.

The scheduler still decides which ready process runs next.

A scheduling example

Event	Process A	Process B
A executes	Running	Ready
A requests disk data	Blocked	Running
Disk completes; B continues	Ready	Running
Scheduler selects A	Running	Ready

Assume one core and two processes. I/O completion need not preempt B immediately.

Check your understanding

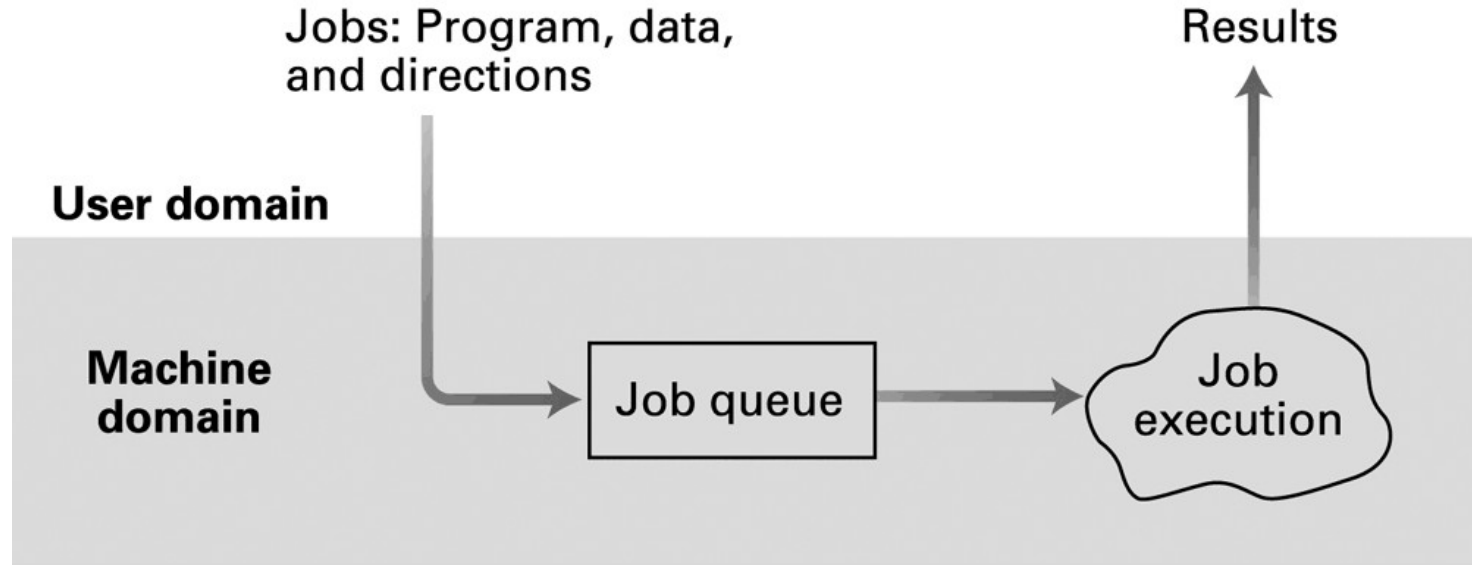
A car awaits a part. Another is ready. Must the mechanic remain idle?

Two mechanics repair two cars at once: concurrency, parallelism, or both?

What must be remembered when a mechanic returns to an unfinished job?

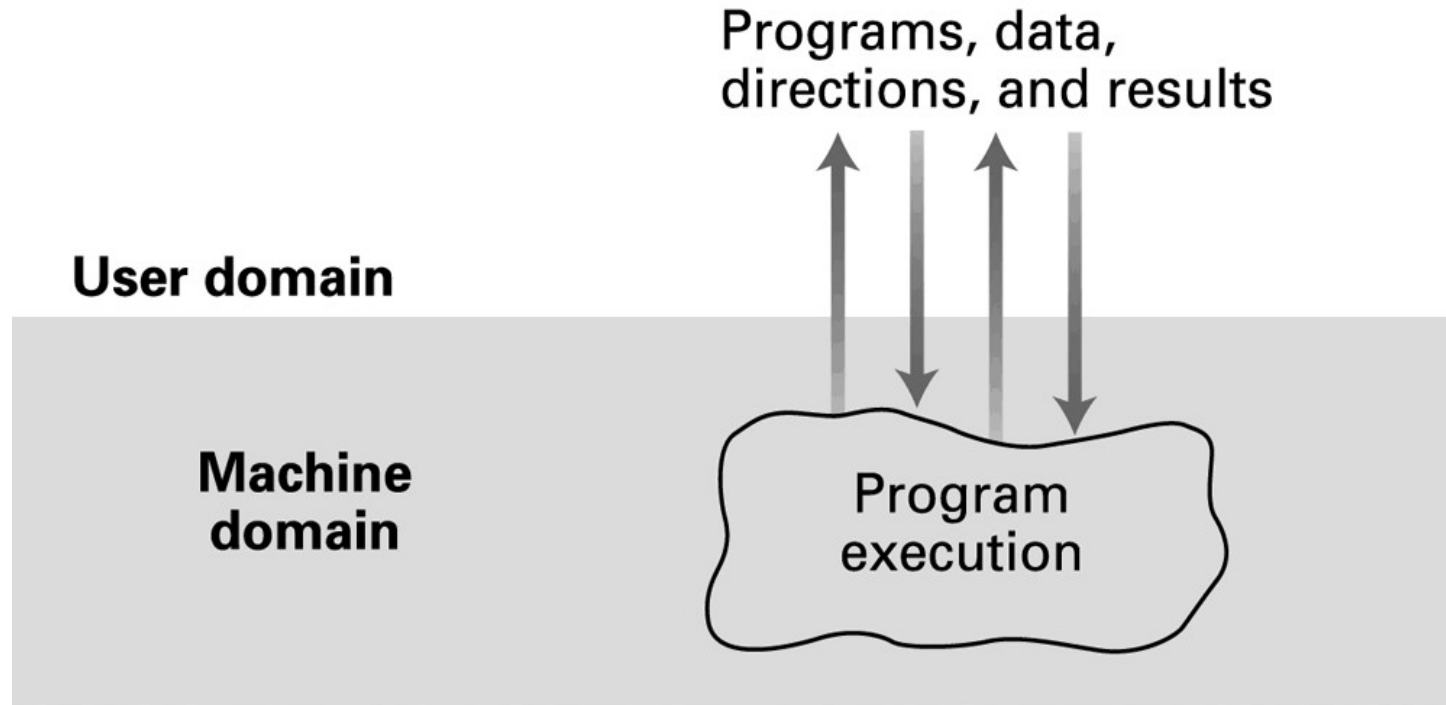
Answers: no; both; the job's progress and working state.

Batch processing



Submit jobs and collect results later; no ongoing conversation with each job.

Interactive processing



Users provide input and receive responses while a program is running.

Interactive and real-time systems

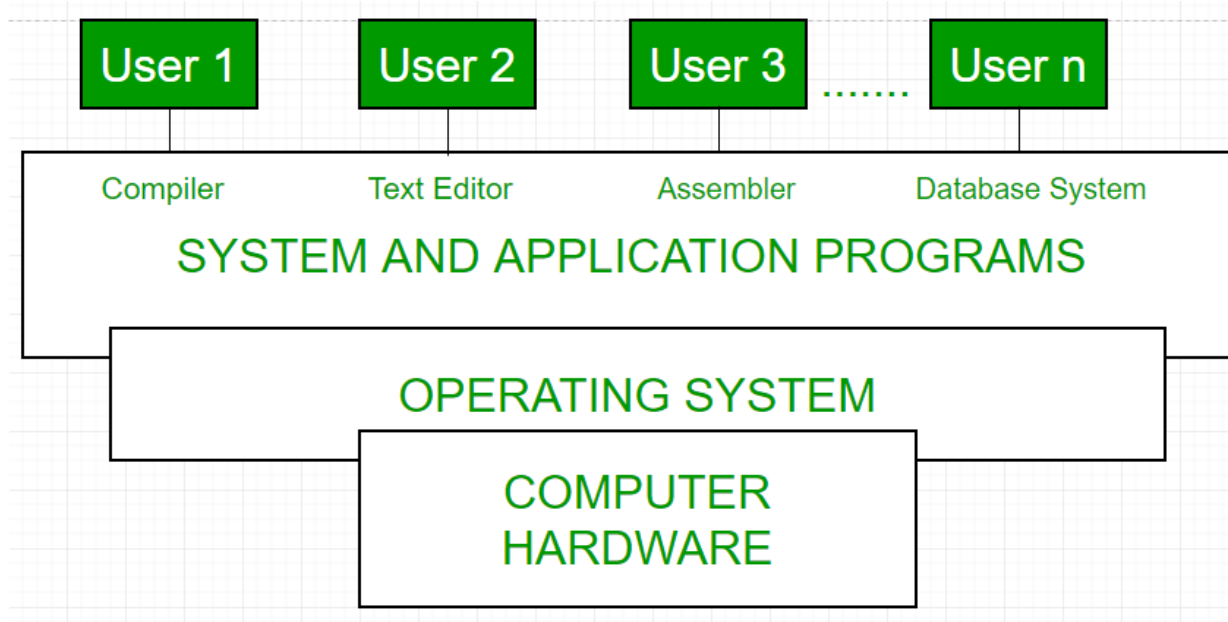
Interactive systems aim to respond promptly to a user.

Real-time systems have timing requirements or deadlines.

A fast response is useful, but speed alone does not make a system real-time.

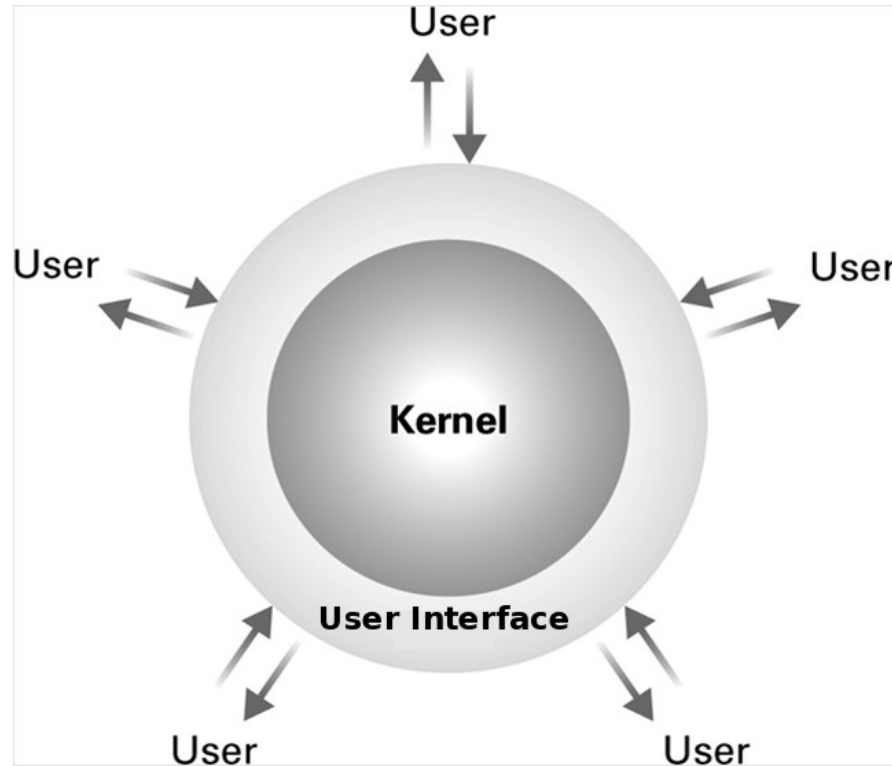
Batch, interactive and real-time describe different requirements.

The operating system's place



Applications request services; the OS manages their use of the hardware.

User interface and kernel



A shell or graphical interface accepts requests. The kernel provides core services.

Files and directories organize persistent data.

Memory management allocates and protects each process's address space.

Device drivers provide software interfaces to particular hardware.

These services support processes while the CPU executes their instructions.

Starting the system

Firmware runs when the machine starts.

A boot mechanism loads an operating-system kernel into memory.

The kernel initializes hardware support and starts system services.

Applications can then run under the operating system's control.

Protection between processes

An ordinary application should not overwrite another process's memory.

Privileged operations are restricted to trusted system code.

Applications request controlled services through the OS.

Sharing hardware requires protection as well as scheduling.

Ideas to retain

Processes have state; scheduling shares CPU time among ready processes.

Waiting for an event is different from waiting for CPU time.

Multiple cores can execute work in parallel.

Context switches preserve execution state so a process can resume correctly.