

Unidad 1

Programación de procesos

Programación de servicios y procesos

Curso 2025/2026

Contenidos

- ▶ Conceptos: programas, ejecutables, procesos y servicios
- ▶ Paradigmas de programación multitarea
 - ▶ Programación concurrente
 - ▶ Programación paralela
 - ▶ Programación distribuida
 - ▶ Programación multihilo
- ▶ Gestión y estados de los procesos
- ▶ Comunicación entre procesos
- ▶ Sincronización entre procesos

Conceptos: programa y ejecutable

- ▶ **Programa:** conjunto de instrucciones que ejecutadas en un ordenador realizarán una tarea o ayudarán al usuario a realizarla
- ▶ **Ejecutable** es un fichero que contiene el código binario o interpretado que será ejecutado en un ordenador

Conceptos: proceso

- ▶ **Proceso:** *grosso modo* es el programa en ejecución
 - ▶ Desde el punto de vista del S.O. un proceso es una unidad de trabajo que consume recursos (tiempo de CPU y memoria)
 - ▶ La ejecución de un programa dar lugar a un proceso que a su vez puede arrancar más procesos
 - ▶ Los procesos son independientes: si se ejecuta un programa dos veces se obtienen dos procesos distintos
- ▶ **Servicio:** programa diseñado para ejecutarse como proceso en segundo plano sin interacción directa con el usuario
 - ▶ Los servicios suelen estar en ejecución constante

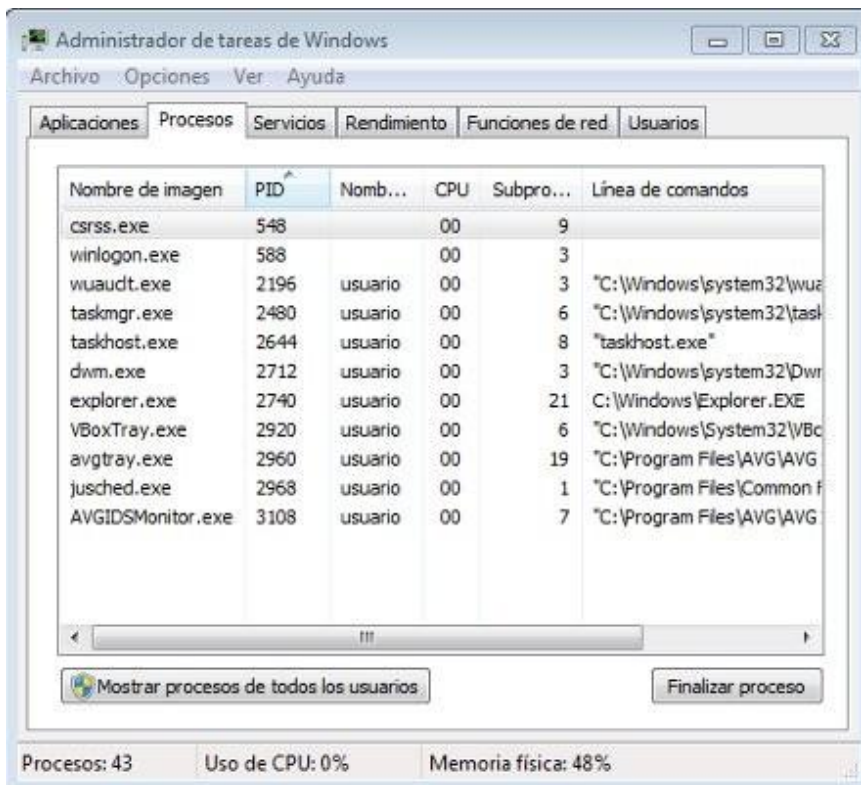
Conceptos: servicio

- ▶ **Servicio:** programa diseñado para ejecutarse como proceso en segundo plano sin interacción directa con el usuario
 - ▶ Los servicios suelen estar en ejecución constante
 - ▶ Esperan peticiones o realizan tareas periódicas
- ▶ Ejemplos:
 - ▶ Windows: Spooler (gestión de impresión), Windows Update
 - ▶ Linux: sshd (servidor de conexiones SSH), cron (planificador de tareas)

Procesos en Windows y Linux

- ▶ Todo proceso está numerado con un **PID** (Process ID)
- ▶ Se pueden ver los procesos de forma gráfica
 - ▶ Windows: administrador de tareas
 - ▶ Linux: Monitor del sistema

Procesos en Windows y Linux



Administrador de tareas de Windows

Archivo Opciones Ver Ayuda

Aplicaciones Procesos Servicios Rendimiento Funciones de red Usuarios

Nombre de imagen	PID	Nomb...	CPU	Subpro...	Línea de comandos
csrss.exe	548		00	9	
winlogon.exe	588		00	3	
wuauclt.exe	2196	usuario	00	3	"C:\Windows\system32\wuauclt.exe"
taskmgr.exe	2480	usuario	00	6	"C:\Windows\system32\taskmgr.exe"
taskhost.exe	2644	usuario	00	8	"taskhost.exe"
dwm.exe	2712	usuario	00	3	"C:\Windows\system32\Dwm.exe"
explorer.exe	2740	usuario	00	21	C:\Windows\Explorer.EXE
VBoxTray.exe	2920	usuario	00	6	"C:\Windows\System32\VBoxTray.exe"
avgtray.exe	2960	usuario	00	19	"C:\Program Files\AVG\AVG\avgtray.exe"
jusched.exe	2968	usuario	00	1	"C:\Program Files\Common Files\AVG\AVG\jusched.exe"
AVGIDSMonitor.exe	3108	usuario	00	7	"C:\Program Files\AVG\AVG\AVGIDSMonitor.exe"

Mostrar procesos de todos los usuarios Finalizar proceso

Procesos: 43 Uso de CPU: 0% Memoria física: 48%



Monitor del sistema

Monitor Editar Ver Ayuda

Sistema Procesos Recursos Sistemas de archivos

Carga media para los últimos 1, 5 y 15 minutos: 0,01, 0,08, 0,21

Nombre del proceso	Estado	% CPU	Prioridad	ID	Memoria	Canal en espera
gnome-system-monitor	Ejecutándose	10	0	6373	7,1 MiB	0
shutter	Durmiendo	0	0	3423	77,1 MiB	poll_schedule_timeout
Xorg	Durmiendo	0	0	1851	41,7 MiB	poll_schedule_timeout
soffice.bin	Durmiendo	0	0	2289	161,2 MiB	poll_schedule_timeout
gnome-panel	Durmiendo	0	0	2136	12,6 MiB	poll_schedule_timeout
nautilus	Durmiendo	0	0	2135	24,4 MiB	poll_schedule_timeout
bash	Durmiendo	0	0	6274	2,9 MiB	n_tty_read
kondemand/0	Durmiendo	0	0	63	N/D	worker_thread
kondemand/1	Durmiendo	0	0	64	N/D	worker_thread
gnome-pty-helper	Durmiendo	0	0	6273	792,0 KiB	unix_stream_data_wait

Finalizar proceso

Manejo de procesos en Windows

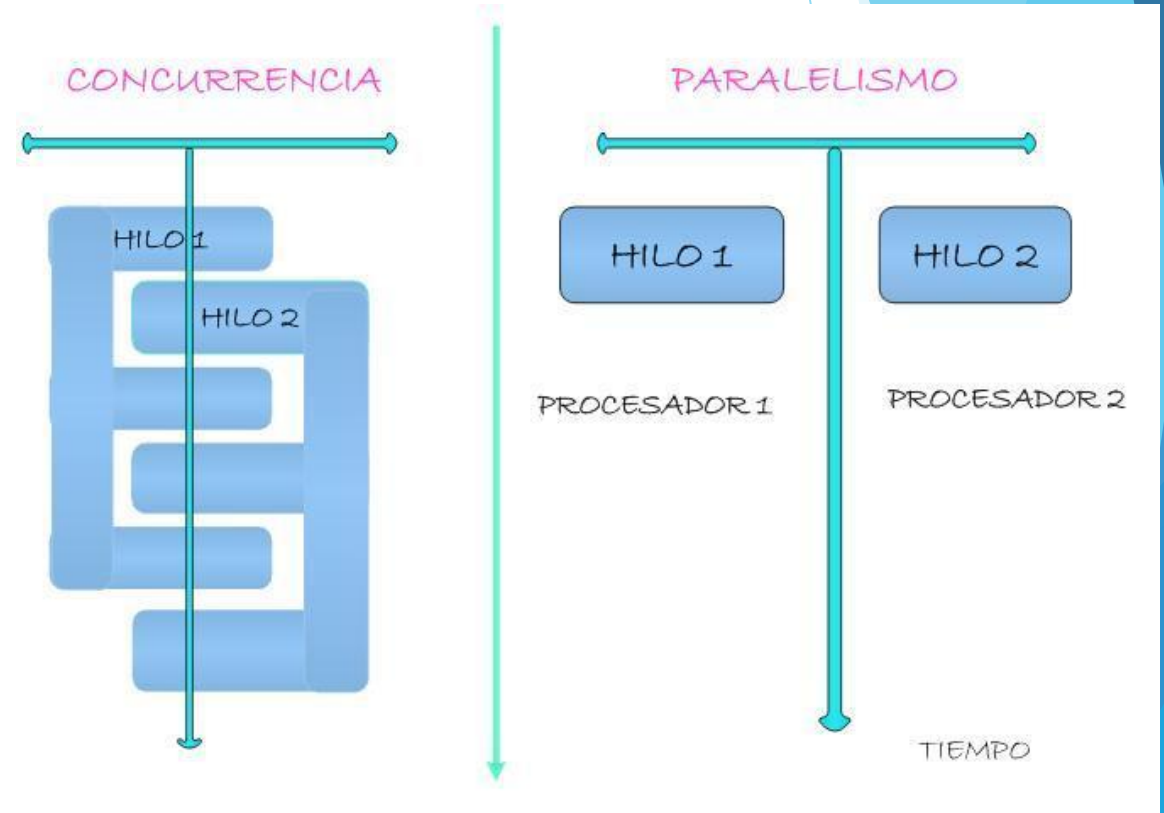
- ▶ En la línea de comandos (CMD o símbolo del sistema):
 - ▶ **tasklist**. Lista los procesos presentes en el sistema, mostrando:
 - ▶ nombre del ejecutable
 - ▶ PID
 - ▶ Uso de memoria
 - ▶ **taskkill** . “Mata” procesos, es decir, termina su ejecución
 - ▶ Con la opción /PID se especifica el proceso a eliminar

Manejo de procesos en Linux

- ▶ **ps.** Lista los procesos presentes en el sistema
 - ▶ Por defecto los que ha lanzado el usuario en esa sesión de shell
 - ▶ Con la opción "aux" muestra todos los procesos del sistema
- ▶ **pstree.** Muestra un listado de procesos en forma de árbol, mostrando qué procesos han creado otros.
 - ▶ Con la opción "AGu" usa líneas guía y muestra el nombre de usuario propietario del proceso.
- ▶ **kill.** Manda señales a los procesos. La señal -9, matará al proceso. Se utiliza "kill -9 <PID>".
- ▶ **killall.** Mata procesos por su nombre. Se utiliza como "killall nombreDeAplicacion".
- ▶ **nice.** Cambia la prioridad de un proceso.
 - ▶ nice -n 5 *comando*: ejecuta *comando* con una prioridad 5.
 - ▶ Por defecto la prioridad es 0. Las prioridades están entre -20 (más alta) y 19 (más baja).

Concurrencia y paralelismo

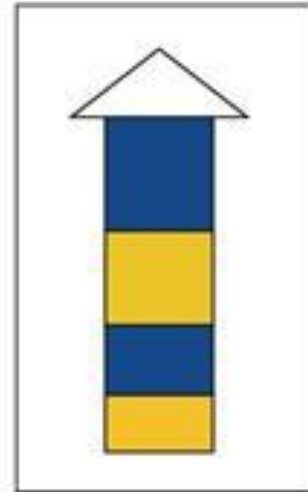
- ▶ Concurrencia: coincidencia de dos tareas (procesos o hilos) en el tiempo
- ▶ La primera instrucción de uno se ejecuta antes que la última instrucción del otro
- ▶ Paralelismo: ejecución simultánea en procesadores distintos
- ▶ En rigor, el paralelismo es concurrencia
 - ▶ La inversa no es cierta



Concurrencia

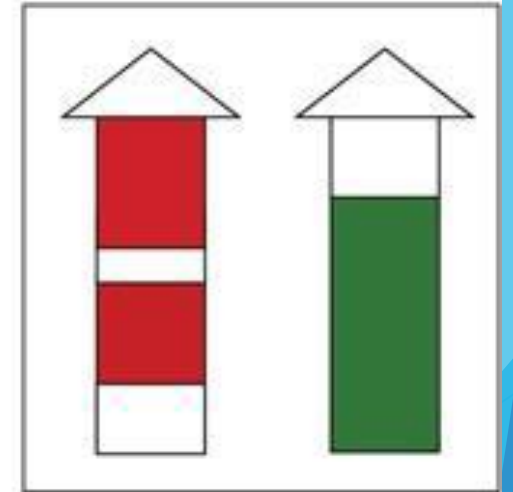
- ▶ La concurrencia se puede realizar en un único procesador, el paralelismo no
- ▶ También se habla de sistemas multitarea:
 - ▶ Multitarea simulada = concurrencia sin paralelismo
 - ▶ Multitarea real = paralelismo

Concurrency



Concurrency is about *dealing with* lots of things at once

Parallelism



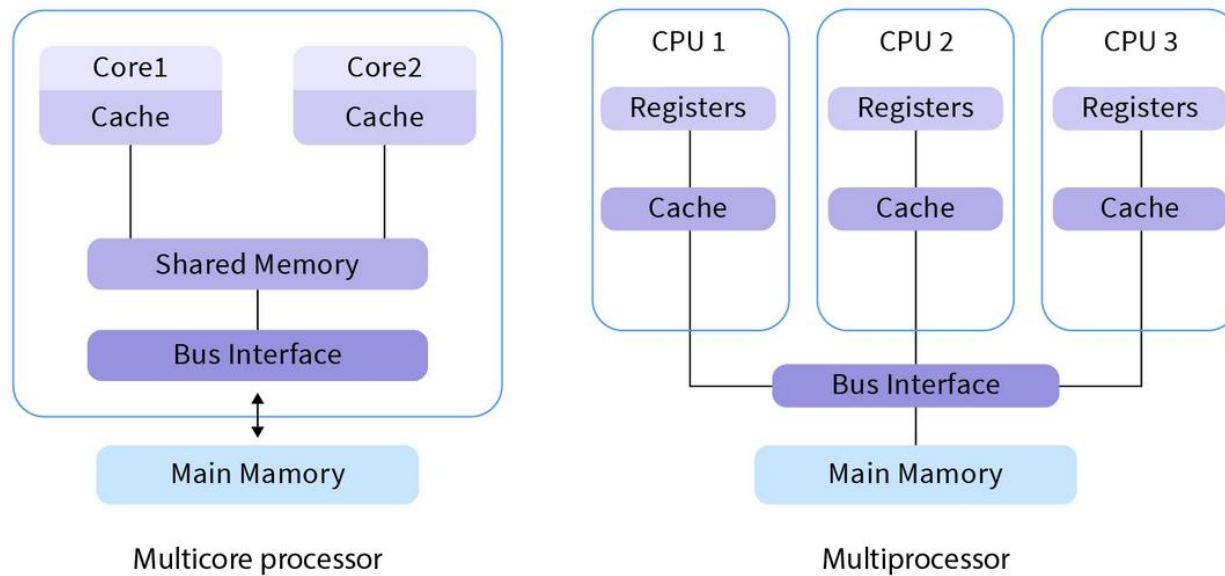
Parallelism is about *doing* lots of things at once

Programación concurrente

- ▶ Programación de varias tareas (procesos o hilos) que se ejecutan de forma solapada en el tiempo
- ▶ Requiere mecanismos de comunicación y sincronización para coordinarse y compartir recursos de forma segura.

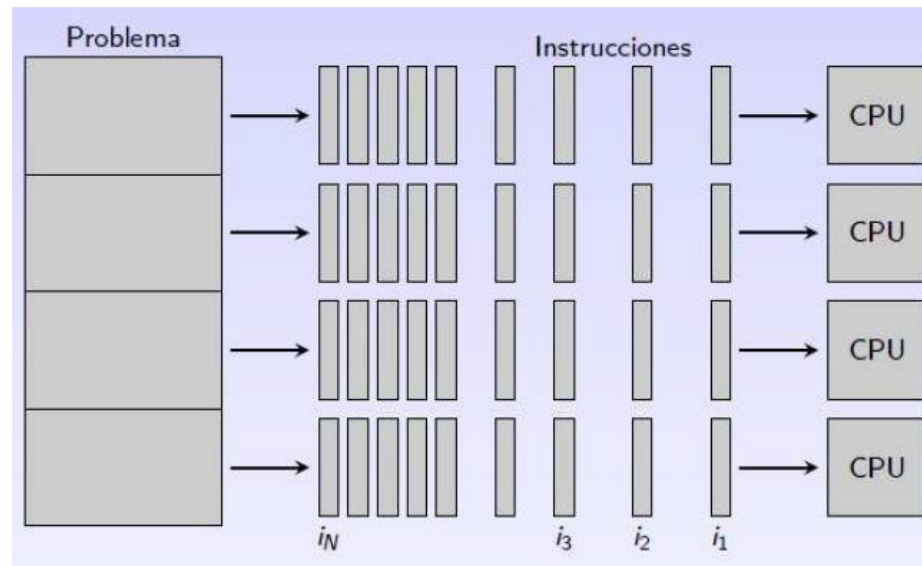
Programación paralela

- Aprovecha sistemas con procesadores multinúcleo o multiprocesadores



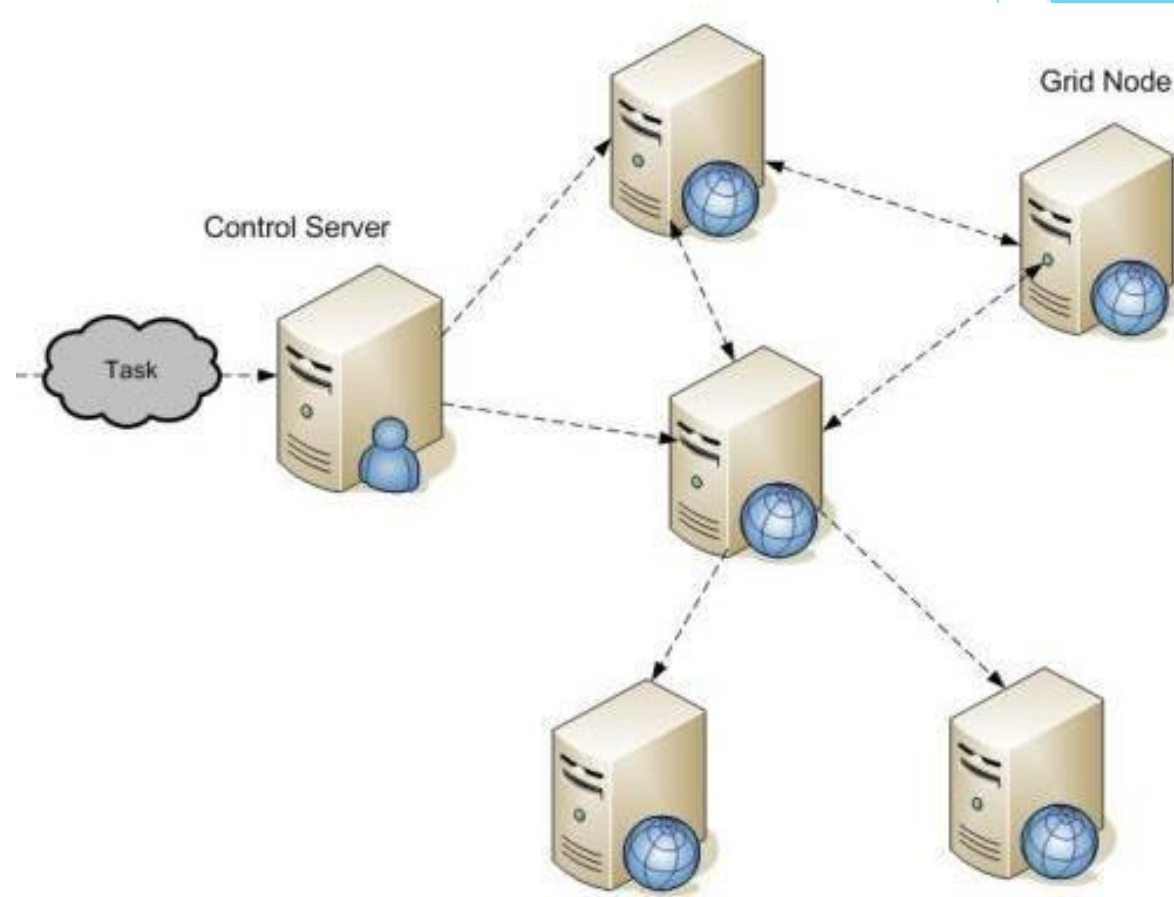
Programación paralela

- Consiste en dividir un problema en subproblemas que puedan ejecutarse de manera simultánea en múltiples procesadores o núcleos, para reducir el tiempo total de ejecución.



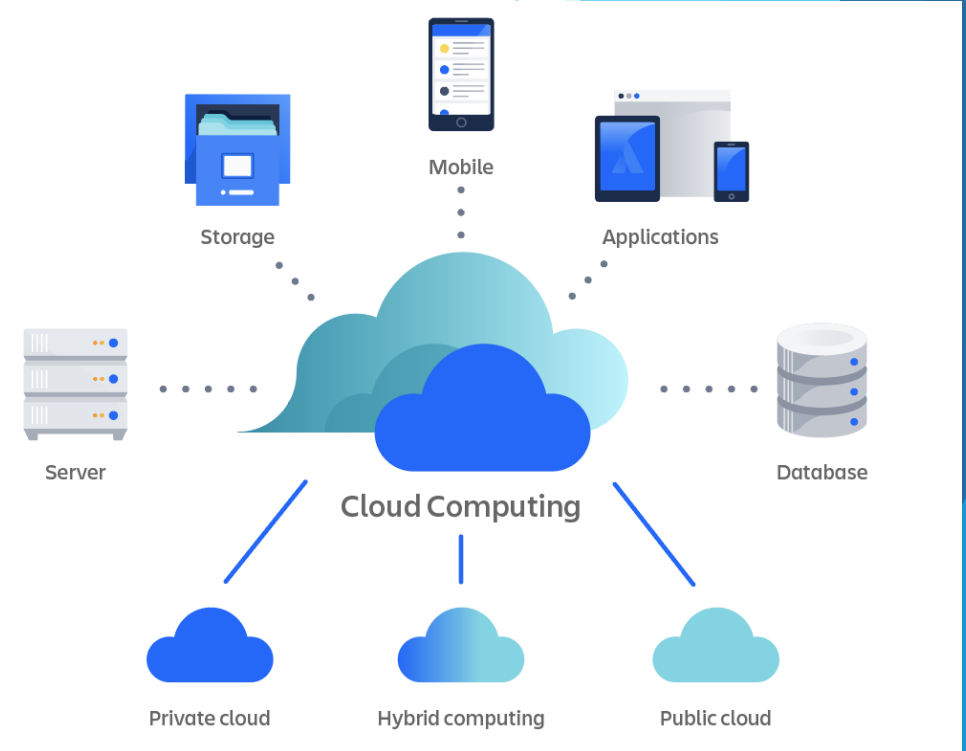
Programación distribuida

- ▶ Ejecución en varios ordenadores conectados por una red
- ▶ La red puede ser de muchos tipos
- ▶ Red creada específicamente para computación distribuida
- ▶ Objetivos:
 - ▶ Alto rendimiento y/o
 - ▶ Alta disponibilidad



Computación en la nube o cloud computing

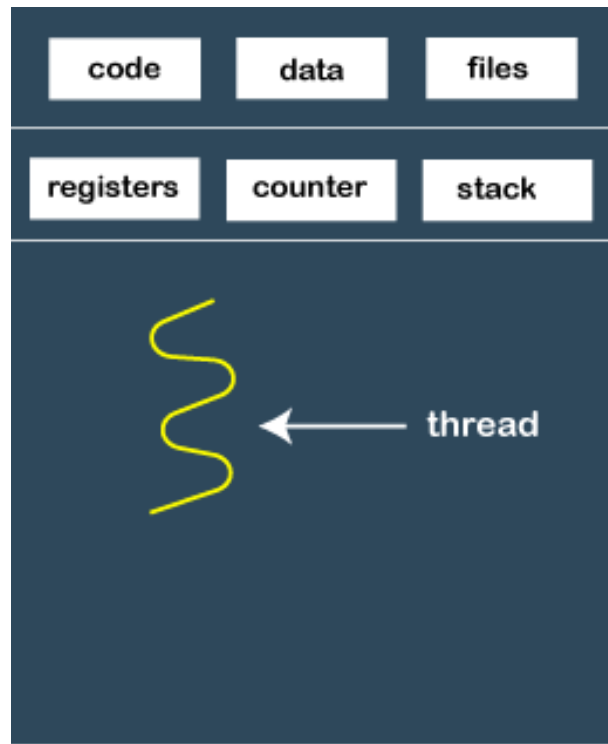
- ▶ Sería un caso particular de programación distribuida
- ▶ Arquitectura cliente-servidor.
- ▶ El usuario (cliente) sólo necesita conexión a Internet.
- ▶ Contratación de servicios y recursos.
- ▶ Recursos gestionados de forma centralizada (p.ej. Amazon, Azure, etc.).
- ▶ Alta escalabilidad.



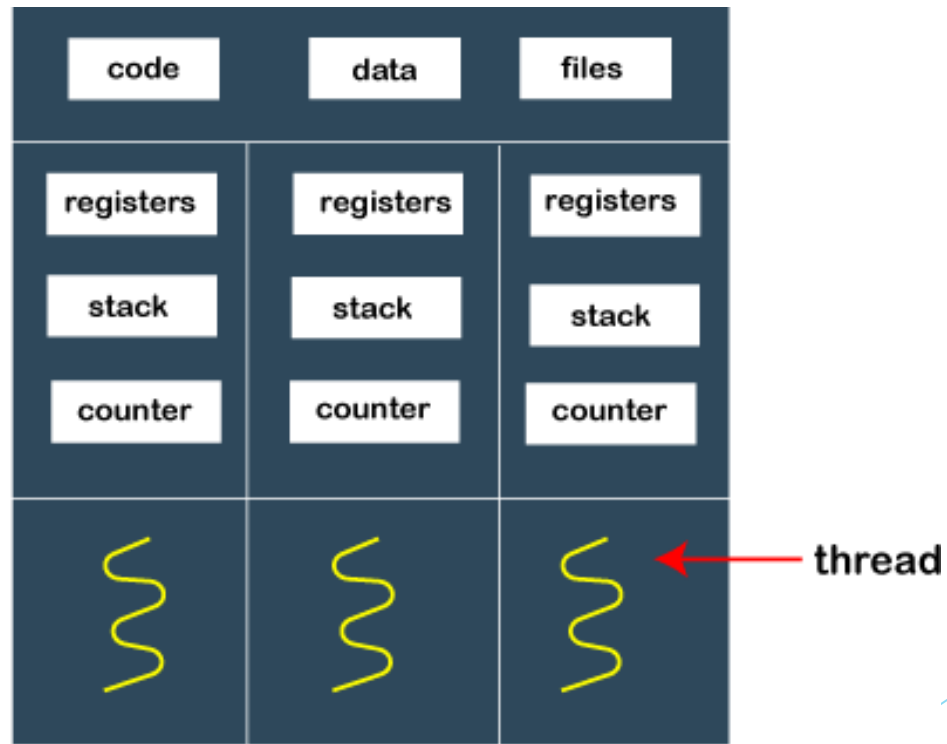
Programación multihilo

- ▶ Limitaciones de los procesos:
 - ▶ Cada proceso tiene su propio espacio en memoria (20 procesos, 20 veces más memoria)
 - ▶ Un proceso no tiene acceso a los datos de otro proceso
- ▶ Por ello se creó la **programación multihilo**
 - ▶ Un **hilo** o **thread** es una unidad de ejecución dentro de un proceso
 - ▶ Creación de múltiples hilos es más rápido que crear múltiples procesos
 - ▶ La comunicación es fácil entre hilos: comparten memoria
 - ▶ Sincronización más compleja

Programación multihilo



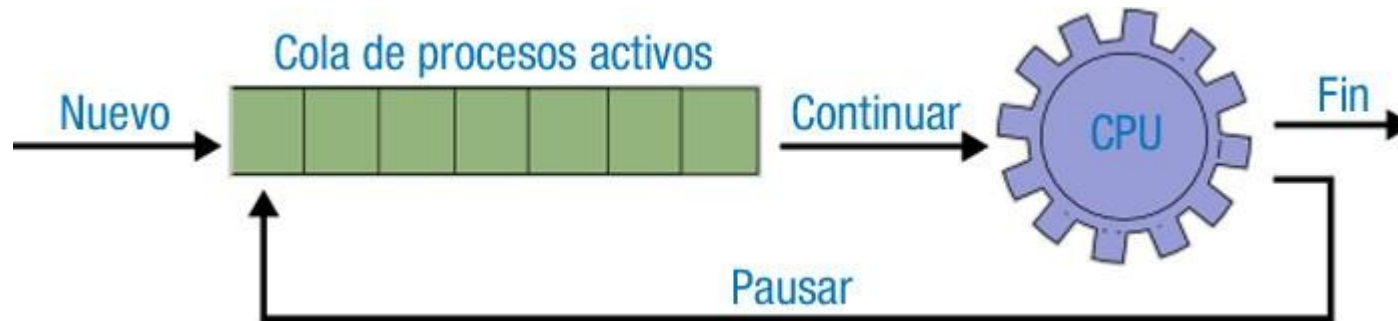
Single-threaded process



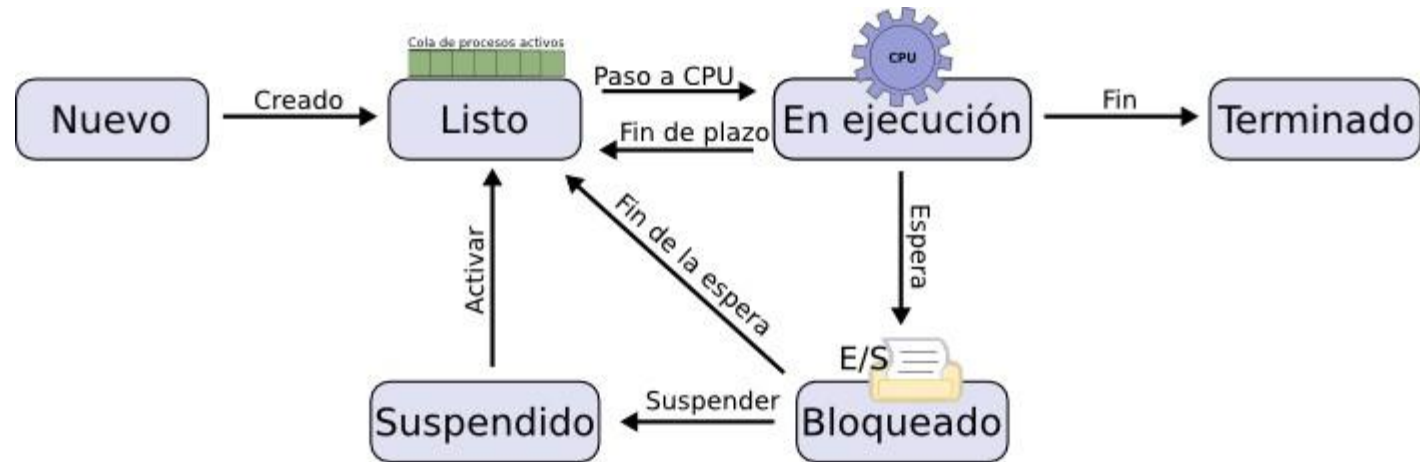
Multi-threaded process

Gestión y estado de los procesos

- Esquema simplificado:

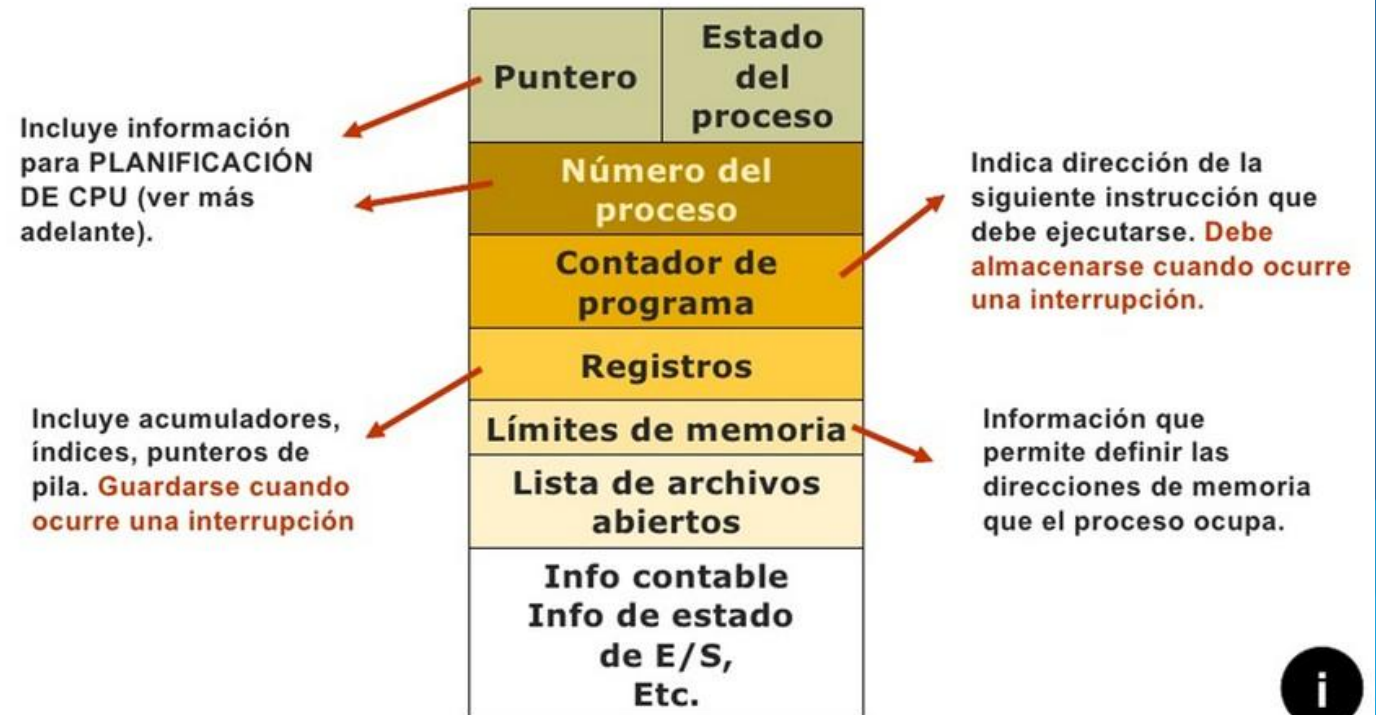


Estados de un proceso



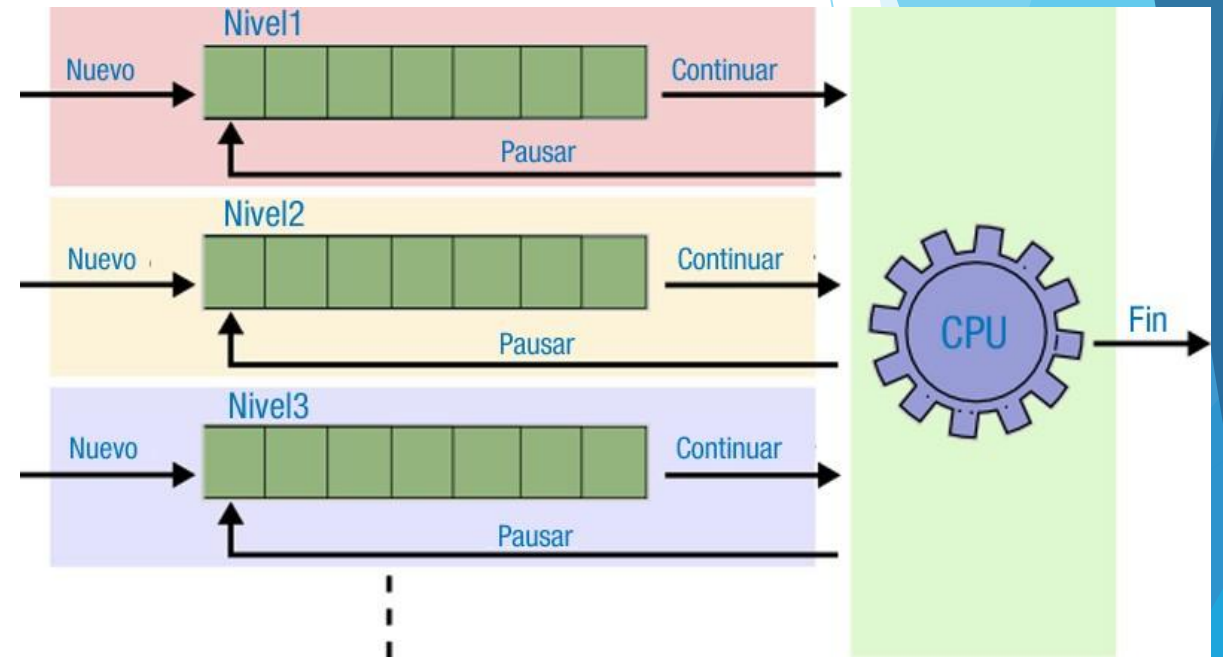
Bloque de control de procesos

- ▶ PCB (Process Control Block)
- ▶ Almacena toda la información necesaria para la ejecución de un proceso
- ▶ Incluye información para que el SO planifique cuando ejecutarlo
- ▶ Se actualiza cuando se pausa un proceso (cambio de contexto)



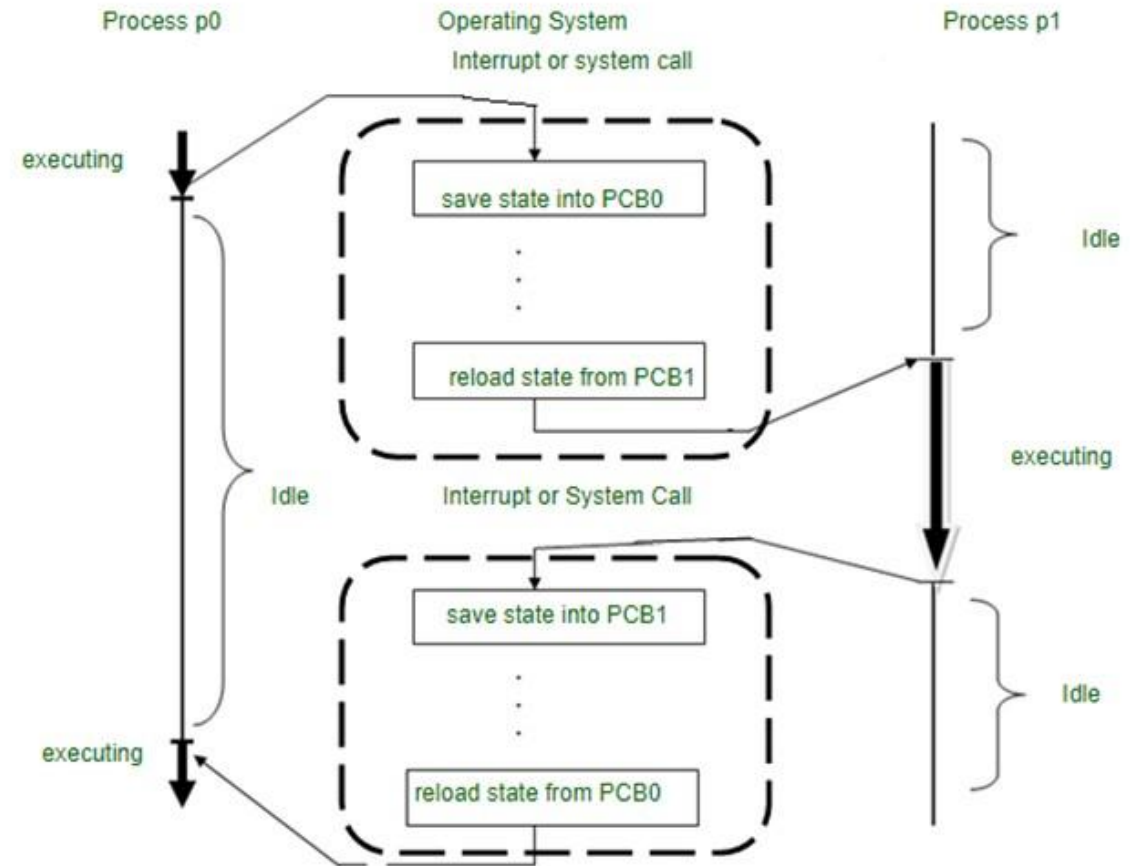
Planificador de tareas

- ▶ Varias colas de procesos por prioridades
- ▶ Algoritmos rotatorios no equitativos
 - ▶ Los procesos críticos (máxima prioridad) deben ejecutarse antes que cualquier otro
- ▶ Los procesos en ejecución se encolan cuando han excedido un tiempo de ejecución llamado *quantum*



Cambio de contexto

- ▶ Almacenamiento de información en PCB del proceso en pausa
- ▶ Recuperación de información de PCB del proceso que pasa a ejecución



Comunicación entre procesos

- ▶ Varios métodos:
 - ▶ **Streams (I/O estándar)**
 - ▶ Lo más habitual entre procesos padres e hijos
 - ▶ **Sockets**
 - ▶ Muy común en aplicaciones cliente-servidor
 - ▶ En la misma máquina o en máquinas distintas
 - ▶ **Ficheros compartidos**
 - ▶ No muy eficiente ni seguro

Sincronización entre procesos

- ▶ Soporte potente en C (semáforos, mutex, memoria compartida...)
- ▶ En Java no hay soporte nativo
 - ▶ La multitarea en Java es principalmente multihilo
 - ▶ Sí hay métodos para controlar la ejecución de procesos:

Mecanismo	Clase	Método
Ejecución	Runtime	exec()
Ejecución	ProcessBuilder	start()
Espera	Process	waitFor()
Generación de código de retorno	System	exit(valor)
Eliminación de procesos	Process	destroy()