EDITORIAL SINTESIS

Programación de servicios y procesos

Capítulo 1: Programación de procesos

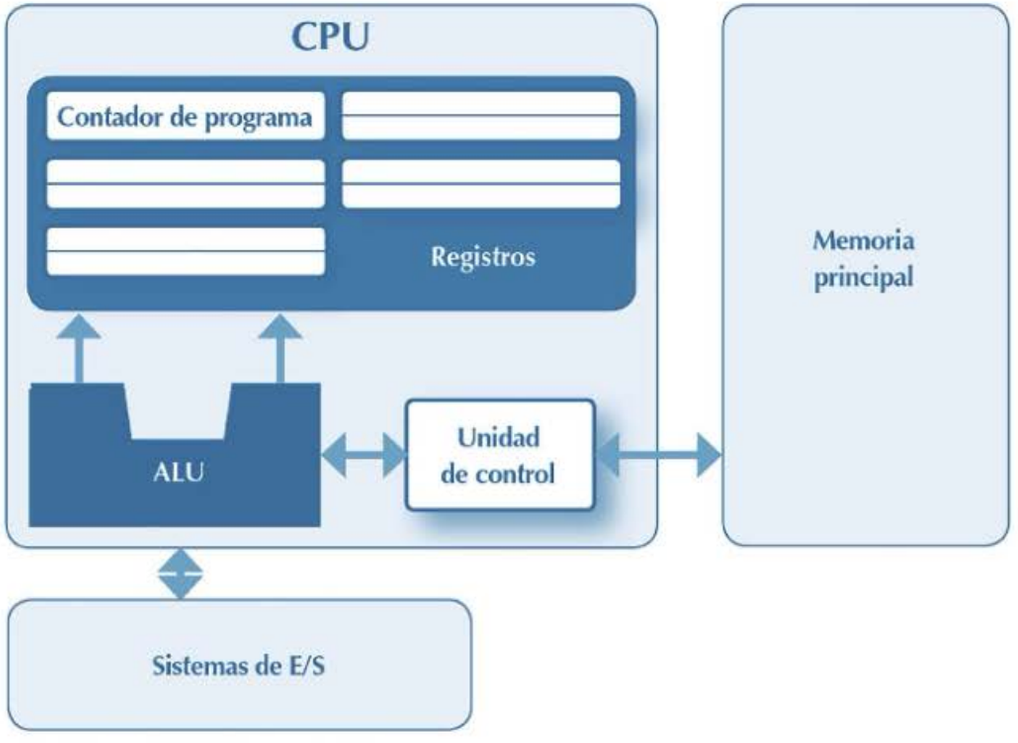
Carlos Alberto Cortijo Bon

- Programa: Conjunto de instrucciones que se pueden ejecutar directamente en una máquina.
  - Es un objeto estático. No cambia. Se suele almacenar en un fichero binario en un medio de almacenamiento secundario.
  - Un programa en Java se ejecuta en una máquina virtual, no física.
- Proceso: Instancia de un programa en ejecución.
  - Es un objeto dinámico. Cambia de estado durante su ejecución.
  - Toda la información relativa a un proceso se almacena en memoria.

Se crean procesos:

- Al comenzar la ejecución de un programa.
- Dinámicamente. Un proceso en ejecución puede crear nuevos procesos sobre la marcha.

- Un proceso se ejecuta en un procesador o CPU.
- Para crear un proceso, previamente, se carga un programa en memoria.
- Un proceso utiliza recursos del sistema:
  - Memoria. Para el propio programa y para datos.
  - Dispositivos de entrada/salida.



Ejecución secuencial de procesos

CPU	A	A	A	A	A	B	B	B	B	C	C	C	C	C	C
Tiempo →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

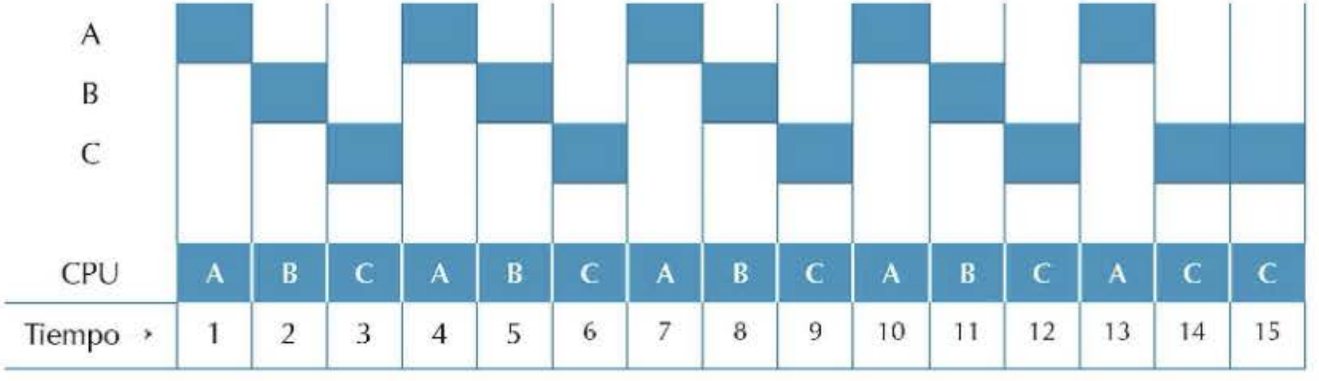
- Los procesos se ejecutan uno tras otro, secuencialmente.

Paralelismo (en un sistema multiprocesador)

CPU 1	A	A	A	A	A										
CPU 2	B	B	B	B											
CPU 3	C	C	C	C	C	C									
Tiempo →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- Ejecución simultánea de los procesos, cada uno en un procesador.

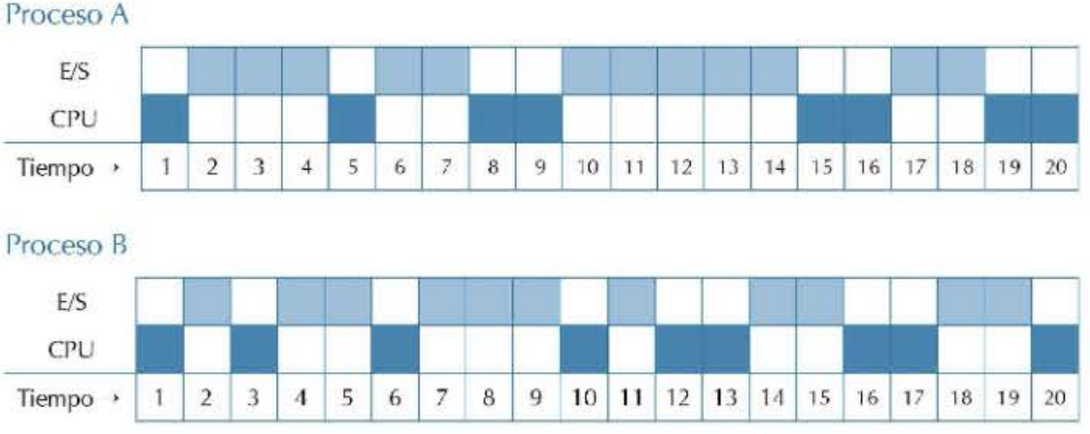
Multitarea (1)



- El tiempo de procesador se reparte entre los distintos procesos.
- Cambio de contexto del procesador cada vez que se pasa a ejecutar un nuevo proceso.

Multitarea (2)

- Las operaciones de entrada/salida son muy lentas.
- Un proceso puede pasar gran parte de su tiempo de ejecución esperando a que se realicen, y en ese tiempo no utiliza el procesador.



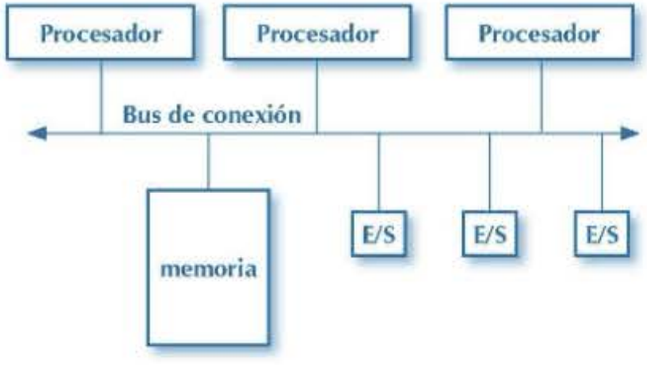
- El tiempo de procesador se reparte entre los distintos procesos.
- Cambio de contexto del procesador cada vez que se pasa a ejecutar un nuevo proceso.

- Cuando un proceso realiza una operación de E/S, se hace un cambio de contexto.
- Así se aprovecha al máximo el procesador.

Multitarea con procesos A y B en un procesador

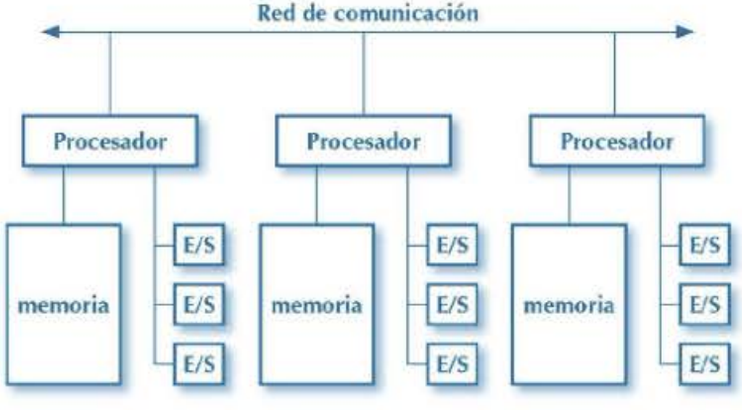
E/S		A	A	A		A	A			A	A	A	A	A			A	A			
			B		B	B		B	B	B		B			B	B			B	B	
CPU	A	B		B	A		B	A	A		B		B	B	A	A	B	B	A	A	B
Tiempo →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

Sistemas multiprocesadores

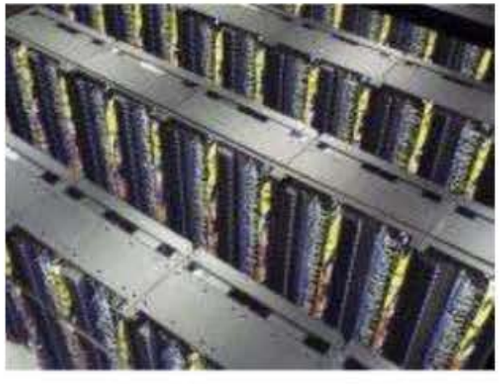


Fuertemente acoplado.

Ejemplo: SMP
(multiprocesamiento simétrico).



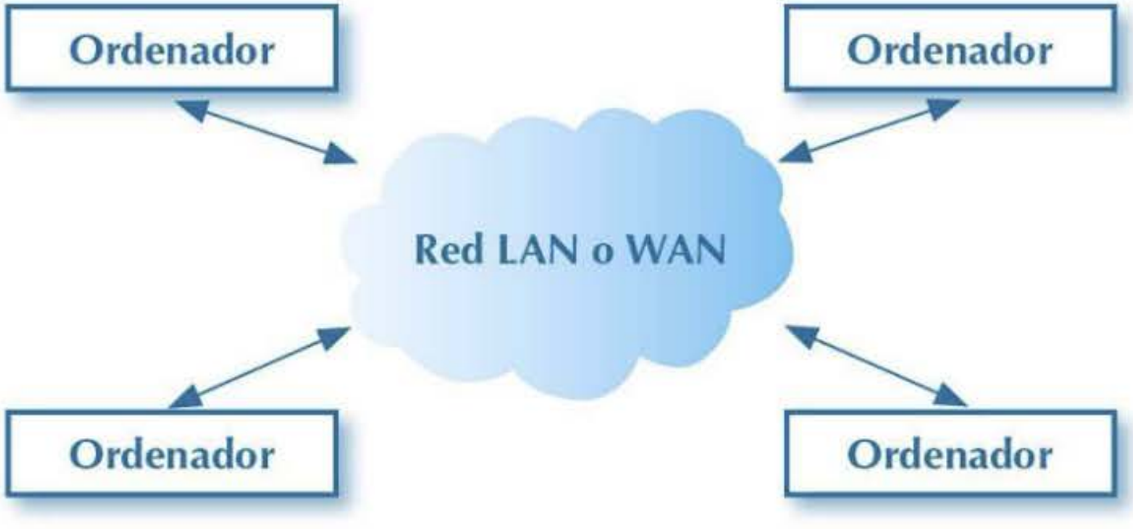
Débilmente acoplado.



Ejemplo: Supercomputador
Red de comunicación de alto rendimiento.

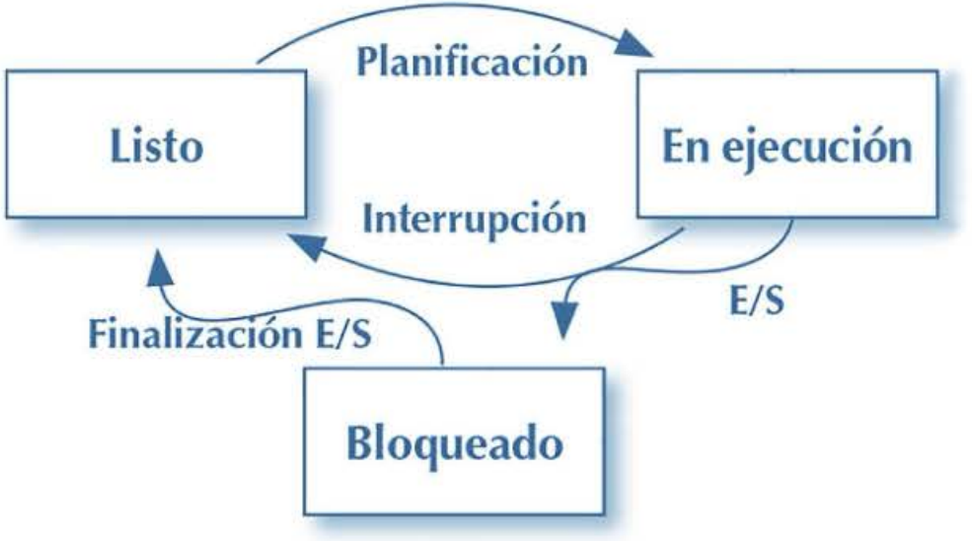
Ejemplo: Cluster Beowulf
Red de comunicación: red de área local convencional.

Sistemas distribuidos



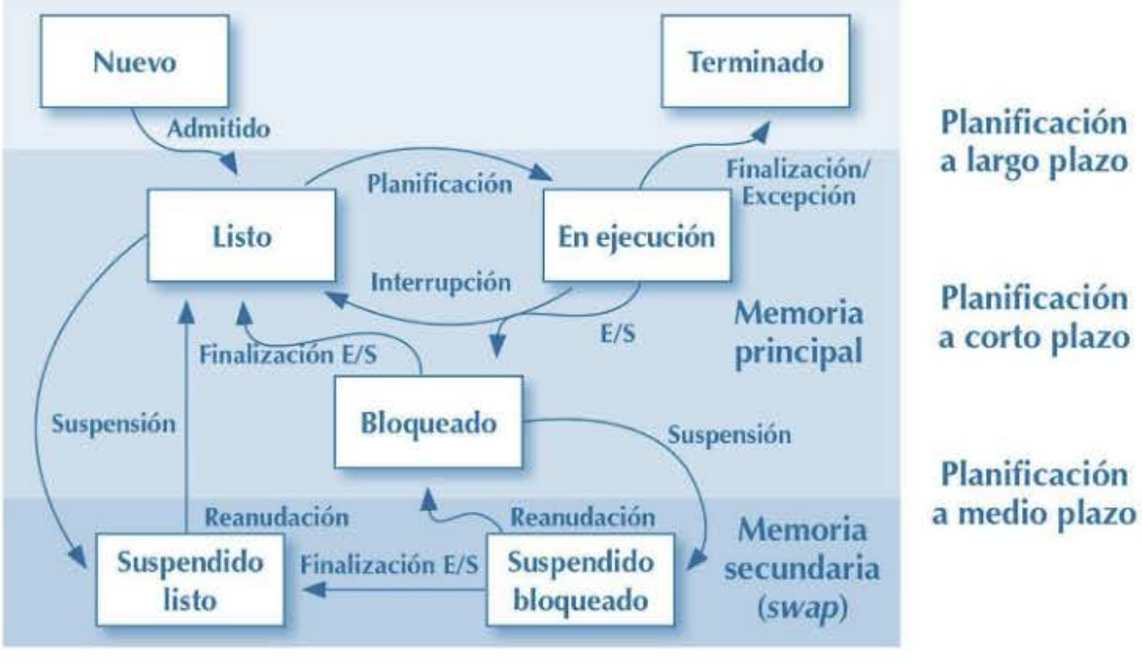
- Sistemas heterogéneos.
- Utilizan protocolos estándares de red para comunicación: TCP o UDP sobre IP.

Planificación de procesos a corto plazo



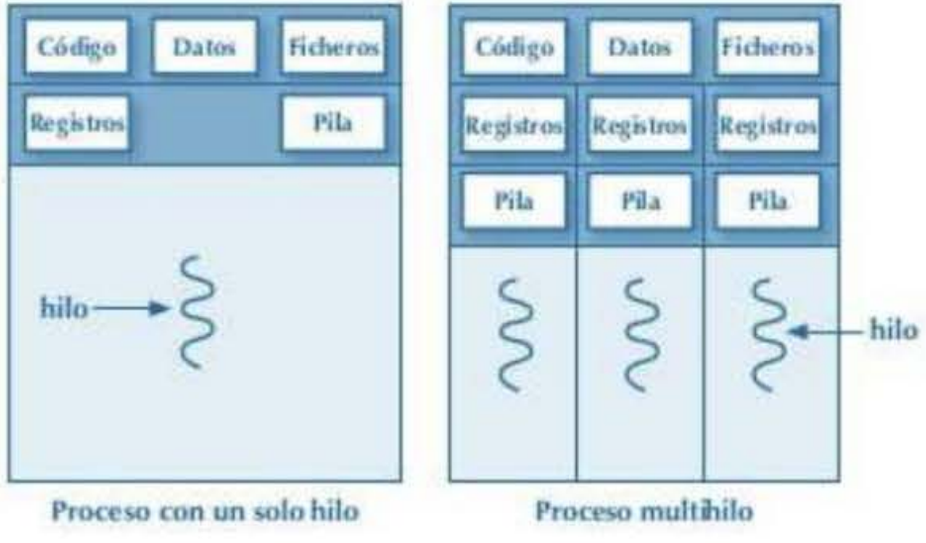
- La realiza el planficador (*scheduler*) del núcleo (*kernel*) del sistema operativo.
- Una rutina de tratamiento de interrupción invoca periódicamente al planificador, que decide cuándo hay que realizar un cambio de contexto para pasar a ejecutar un nuevo programa.

Planificación de procesos a medio y largo plazo



- Planificación a corto plazo: procesos en memoria principal.
- Planificación a medio plazo: paso a y desde memoria virtual.
- Planificación a largo plazo: admisión/terminación de procesos.

Procesos e hilos



- La ejecución de un proceso comienza con un hilo.
- Se pueden crear más hilos sobre la marcha.
- Los hilos comparten memoria (código y datos) y ficheros.
- Cada hilo tiene su propio estado de ejecución (registros y pila). El planificador a corto plazo gestiona cada hilo por separado y realiza cambios de contexto entre hilos.

Servicios

- Tipo particular de procesos: procesos servidores.
- Se ejecutan en segundo plano: no muestran interfaz de usuario.
- Proporcionan servicios a otros procesos.
- Pueden crear nuevos hilos para atender a más procesos simultáneamente: servidores multihilo.

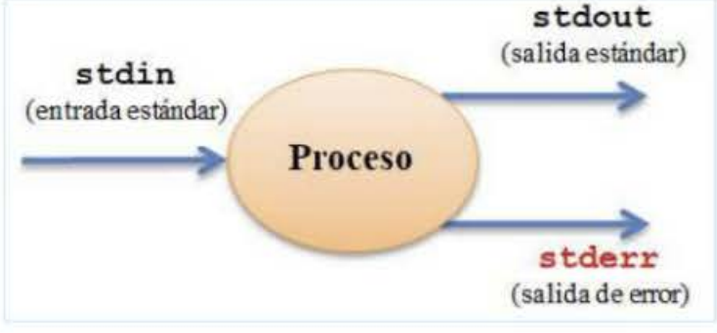
Clases e interfaces para gestión de procesos en Java

- `Runtime`. Entorno de ejecución (JVM o máquina virtual de Java).
  - **`getRuntime`**. Método estático que devuelve Runtime para máquina virtual de Java.
  - **`exec`**. Crea y ejecuta un proceso.

- `ProcessBuilder`. Para crear objetos de clase `Process` (abstracta).
  - **`start`**. Crea y ejecuta un proceso.

- `Process`. Representa un proceso que se ejecuta en la JVM. Es abstracta.
  - **`waitFor`**. Espera a que termine la ejecución del proceso.
  - **`exitValue`**. Devuelve valor de salida o código de retorno.
  - **`destroy`**, **`destroyForcibly`**: Terminan proceso.
  - **`isAlive`**. Averigua si el proceso ha comenzado y no ha terminado su ejecución.

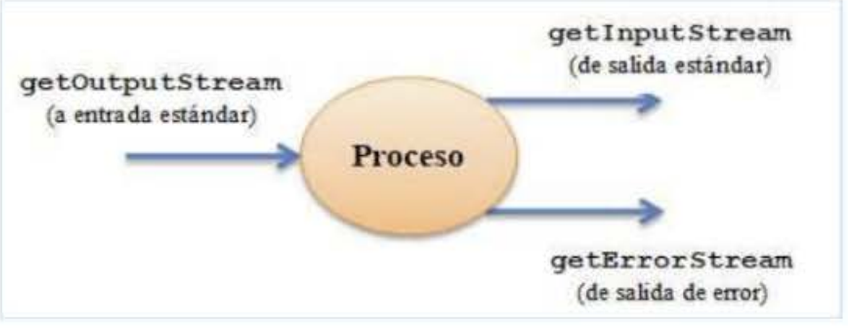
Redirección de entrada y salida



La clase `ProcessBuilder` tiene diversos métodos para redirigir la entrada estándar y salida estándar y de error hacia/desde:

- Las del proceso padre (entrada y salida).
- Un fichero (entrada y salida estándar). Si se dirige la salida hacia un fichero, se pueden sobrescribir sus contenidos o se puede añadir al final.
- El desecho (salida). Se descarta la salida sin más.
- La salida estándar de un proceso hacia la entrada estándar de otro (mediante el mecanismo estándar de tuberías de Linux).

Redirección hacia/desde *streams*



La clase `Process` tiene métodos que permiten obtener:

- Un *stream* de entrada asociado a la salida estándar del proceso (`getInputStream`).
- Un *stream* de entrada asociado a la salida de error del proceso (`getErrorStream`).
- Un *stream* de salida asociado a la entrada estándar del proceso (`getOutputStream`).