UNIT 1: PROCESS PROGRAMMING

Programación de Servicios y Procesos

PROGRAMAS Y PROCESOS

Programa

- Un **programa** contiene un conjunto de **instrucciones** que se pueden **ejecutar** directamente en una **máquina**. Esta máquina puede ser:
  - **Física**
  - **Virtual**: por ejemplo, en el caso del lenguaje Java.
- Un programa es un **objeto estático**, normalmente almacenado en un fichero binario en una memoria secundaria.

PROGRAMAS Y PROCESOS

Proceso

- Corresponde a una **instancia** de un **programa en ejecución.** La ejecución de un programa comienza con la creación y ejecución de un proceso.
- Un proceso **puede crear nuevos procesos** durante su ejecución, que pueden, a su vez, crear nuevos procesos. De esta forma, a partir de un proceso inicial puede crearse una **jerarquía de procesos**.
- Un proceso **no** puede **operar directamente** con los contenidos de la **memoria**. Estos deben traerse de ella y cargarse en **registros del procesador.**
- **PCB (Process Control Block):**
    - Toda la información asociada a un proceso se guarda en un **bloque de control de proceso** (PCB o *Process Control Block)*.
    - Si el programa se ejecuta varias veces, se creará cada vez un nuevo proceso, y cada uno tendrá su propio **PCB**.

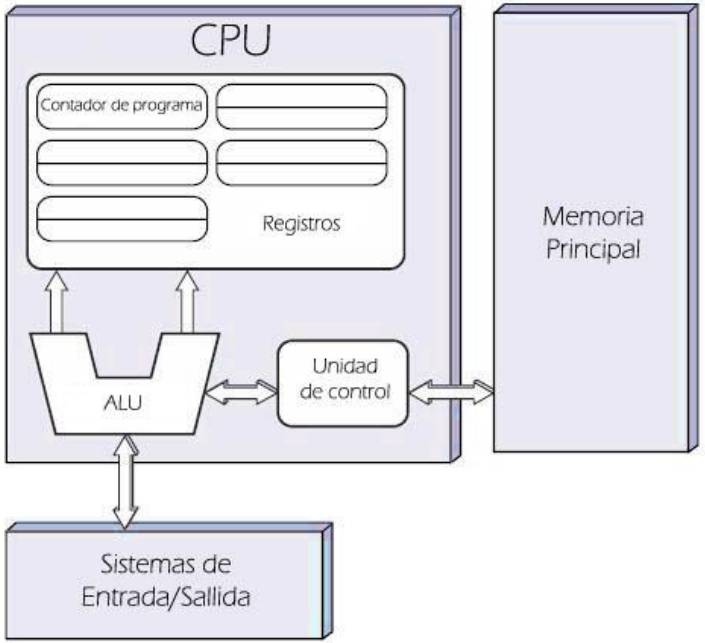
PROGRAMAS Y PROCESOS

Durante su ejecución, un **programa** utiliza diversos recursos del sistema:

- **Memoria principal:** Antes de comenzar su ejecución, un programa debe cargarse en un bloque de la memoria principal, que se asigna al proceso que se crea para ejecutar el programa. Este proceso también puede obtener más memoria, dinámicamente, durante su ejecución.
- **Procesador o CPU** *(Central Processing Unit)***:**
  - **Ejecuta** el proceso una vez **cargado** el programa en **memoria**.
  - El procesador guarda, en un registro especial, el **contador de programa** (PC o *program counter*), la dirección en la memoria de la instrucción que se está ejecutando.
  - Las **operaciones** se realizan en una **unidad aritmético-lógica** (*arithmetic and logical unit* o **ALU**). El resultado se obtiene en un registro desde el que se puede transferir a una posición de la memoria.
- **Dispositivos de entrada/salida (E/S):** Los procesos **comparten** los dispositivos de **E/S**. Debe guardarse **información** acerca de a qué **procesos** se les ha otorgado **acceso** a un dispositivo de **E/S** y del estado de las operaciones realizadas sobre él.

PROGRAMAS Y PROCESOS

Estructura de un procesador:



PROGRAMAS Y PROCESOS

Procesos en Linux:

- En Linux se puede ver la jerarquía de procesos con el comando **pstree**.
- Se puede mostrar información acerca del uso que hacen de la memoria y procesador con el comando **top**.

Procesos en Windows:

- En Windows se puede ver la jerarquía de procesos con **Process Explorer**, programa que suple carencias y defectos del administrador de tareas.
- Más información y descarga de Process Explorer:

https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer

MULTITAREA

Los **sistemas operativos actuales** permiten ejecutar **múltiples procesos** a la vez:

- Editar un fichero de texto.
- Reproducir un vídeo.
- Copiar ficheros de un directorio a otro.
- Descargar un fichero de internet.

Todo ello aunque exista **un solo procesador** en el sistema. A intervalos regulares de tiempo, la ejecución del proceso en curso se detiene y toma el control un programa especial del sistema operativo, que puede parar la ejecución del proceso actual y pasar a ejecutar otro proceso.

La **multitarea** (***multitasking***) consiste en la ejecución simultánea de más de un proceso en un procesador a lo largo de un intervalo de tiempo.

MULTITAREA

En los siguientes ejemplos sólo se está ejecutando un proceso cada vez. En la multitarea, al ser muy rápida la alternancia entre los procesos, a ojos del usuario es como si se ejecutaran a la vez.

Antes de ejecutar un nuevo proceso, se **guarda el estado de ejecución** del mismo, para poder retomarlo más adelante, y se carga el siguiente proceso. Esto se conoce como **cambio de contexto**.

Ejecución de procesos sin multitarea

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

Ejecución de procesos con multitarea

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

MULTITAREA

Si disponemos de más de un procesador (**sistema multiprocesador**), el rendimiento del sistema aumenta. En este caso se pueden ejecutar en paralelo varios procesos, uno en cada procesador:

Ejecución de procesos 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

Pero, normalmente, siempre **habrá** muchos **más procesos** que **procesadores**, por lo que también se hará **multitarea** en **cada procesador**.

MULTITAREA

Puede parecer que en caso de disponer de **un solo procesador no compense la multitarea**, por el tiempo que se pierde en los **cambios de contexto**. Esto no es así por los siguientes motivos:

1. **Tiempo dedicado:** El **tiempo** que se dedica al **cambio de contexto es pequeño** en comparación con el que se dedica a la ejecución de procesos. Un **usuario** suele hacer **varias cosas a la vez** y querrá que **todas las tareas** con las que está **avancen**.

2. **Uso del procesador:** Durante la ejecución de los procesos, suele haber **períodos** muy largos de tiempo en que **no hacen uso del procesador**, por estar esperando a que se terminen de realizar **operaciones de E/S**. Ejemplo: un procesador de texto pasa mucho tiempo esperando a que el usuario pulse una tecla.

MULTITAREA

A continuación se muestra la ejecución de dos procesos que alternan periodos de uso del procesador con periodos de espera para que se realicen operaciones de E/S.

El tiempo total de ejecución es sólo ligeramente superior al de cualquiera de ellos por separado:

Proceso A

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

Proceso B

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

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

PROCESOS Y SISTEMAS MONOPROCESADORES Y MULTIPROCESADORES

- **Sistema monoprocesador:** tiene un único procesador.
- **Sistema multiprocesador:** tiene varios procesadores.

En ambos casos se pueden ejecutar varios procesos de manera simultánea.

- **Procesos concurrentes:** Se **ejecutan simultáneamente** durante un intervalo dado de tiempo, ya sea de forma real (sistemas multiprocesadores) o simulada (en sistemas monoprocesadores).
- **Programación concurrente**. Dos definiciones:
    - Se ejecutan varios procesos concurrentes en un sistema.
    - **Técnicas** que permiten **desarrollar programas** que utilizan varios procesos concurrentes que funcionan de forma conjunta y coordinada para realizar una tarea.
    - Si un programa lanza distintos procesos concurrentes para realizar una tarea, estos deben comunicarse y sincronizarse entre sí.

SISTEMAS MONOPROCESADORES

Los **sistemas multiprocesadores** disponen de más de un procesador.

Multiprogramación: La ejecución concurrente de varios procesos en un sistema monoprocesador.

Como hemos visto, la técnica del **cambio de contexto** permite una multitarea efectiva y un mejor aprovechamiento del procesador, aprovechando los tiempos de inactividad o de operaciones de E/S de los procesos.

Los actuales procesadores **multinúcleo** (***multicore***) contienen varios núcleos o unidades de proceso integradas. Por lo que se pueden considerar **un tipo de sistemas multiprocesadores**.

SISTEMAS MULTIPROCESADORES

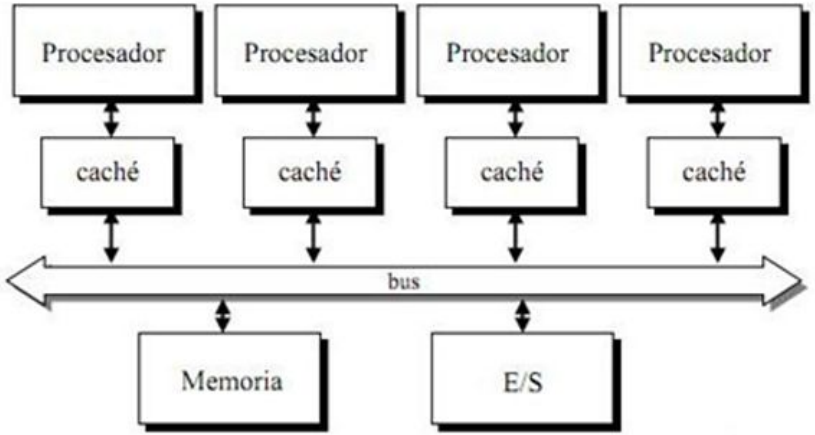
Los **sistemas multiprocesadores** disponen de más de un procesador.

Atendiendo a su arquitectura, se pueden clasificar en:

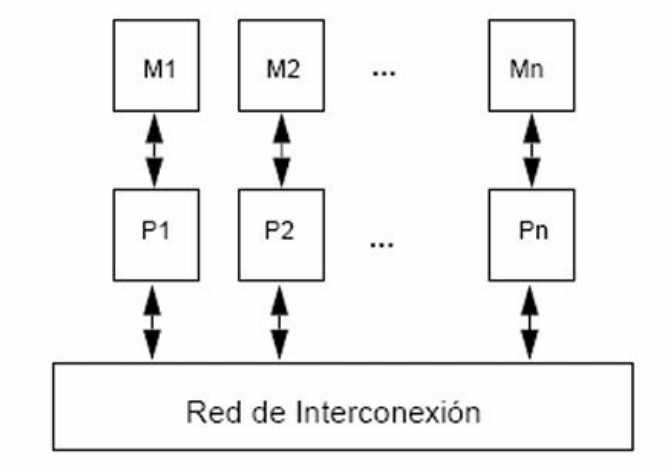
- Sistemas multiprocesadores **fuertemente acoplados: Comparten memoria** a la que acceden a través de un bus de conexión, a partir del cual también acceden al sistema de E/S. Existen dos tipos:
  - *Sistemas multiprocesadores* ***simétricos*** (SMP): Todos los procesadores funcionan de igual manera, no hay ninguno diferenciado del resto.
  - *Sistemas multiprocesadores* ***asimétricos (master-slave)***: Uno de los procesadores del sistema, *master*, controla al resto de procesadores (*slave*).
- Sistemas multiprocesadores **débilmente acoplados:**
  - **No comparten memoria**, cada procesador tiene su propia memoria y su propio sistema de E/S.
  - Existe una **red de comunicaciones** entre los procesadores.
  - Se utilizan en **supercomputación** y para aplicaciones especializadas intensivas en **cálculo**.

SISTEMAS MULTIPROCESADORES

Sistema multiprocesador fuertemente acoplado



Sistema multiprocesador débilmente acoplado



SISTEMAS DISTRIBUIDOS

Los **sistemas distribuidos** son aquellos en que los **procesadores** están en **ordenadores independientes** comunicados mediante una red de área local (LAN) o incluso de área amplia (WAN) o a través de Internet.

Se caracterizan por la **heterogeneidad del hardware** sobre el que están construidos y por la relativa independencia entre los sistemas que los constituyen.

SISTEMAS DISTRIBUIDOS

Programación distribuida: Tiene dos posibles significados:

- La **ejecución** de **varios procesos concurrentes** en un sistema **distribuido**.
- **Desarrollo de programas** que utilizan varios procesos concurrentes que pueden ejecutarse en distintos procesadores de un sistema distribuido, y a las **técnicas** que lo hacen posible.

En un sistema distribuido **no** hay **memoria compartida ni red de conexión específica** entre los distintos ordenadores.

La **comunicación entre procesos** se realiza mediante **mensajes** a través de la red de comunicaciones que comunica los distintos ordenadores, para los que se suelen utilizar los protocolos de red estándares **TCP o UDP.**

Ventaja: son altamente escalables y configurables.

Inconvenientes:

- Sistemas **heterogéneos** y **complejos** de **mantener**.
- **Sincronización** entre procesos **complicada**.
- Al ser la **comunicación** entre procesos a través de la **red**, ésta puede tener una **velocidad limitada**, convirtiéndose en un **cuello de botella**.

VENTAJAS DE LA PROGRAMACIÓN CONCURRENTE

La **programación concurrente**:

- Permite **incrementar el rendimiento** del sistema, porque los procesadores pasan a ejecutar otros procesos cuando el proceso que ejecutan está desocupado o a la espera de que termine una operación de E/S.
- Hace posible un **menor tiempo de respuesta**, porque el tiempo de procesador se distribuye entre todos los procesos en ejecución.

Lo **ideal** es que una **aplicación** se pueda **descomponer** en procesos independientes que se ejecuten en paralelo con un mínimo de interacción y comunicación entre ellos.

Un **sistema multiprocesador** debe **ser flexible** para permitir distribuir dinámicamente la carga de trabajo entre los procesadores disponibles. También debe ser **escalable** para que la potencia de cálculo aumente con la instalación de nuevos procesadores.

Un **sistema multiprocesador redundante** tiene **procesadores suplementarios** que proporcionan **alta disponibilidad** y **tolerancia a fallos**.

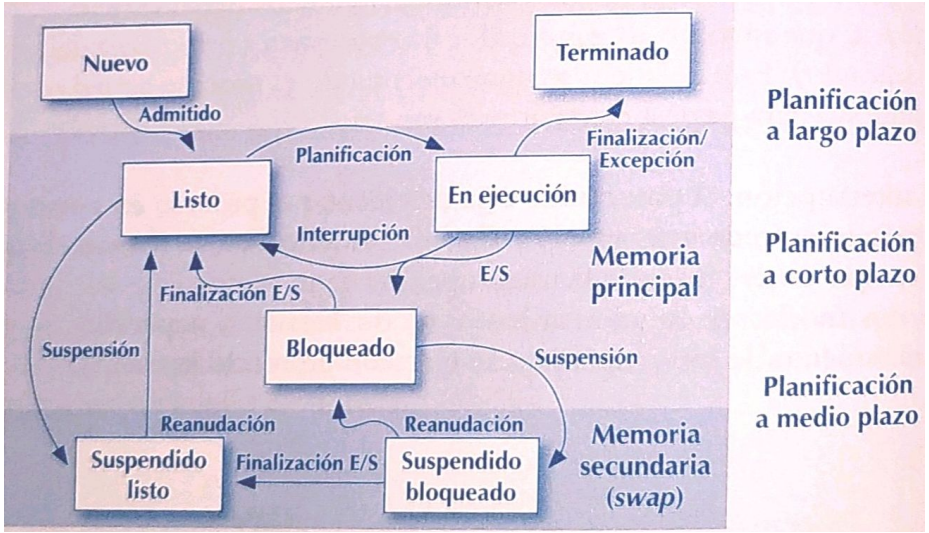
INCONVENIENTES DE LA PROGRAMACIÓN CONCURRENTE

- Los principales inconvenientes pueden venir de la **dificultad** de implementar mecanismos adecuados de **sincronización** y **comunicación** entre **procesos**.
- Según el caso, es posible que la **sobrecarga** que suponen **anule la mejora** en el rendimiento que proporciona la ejecución concurrente.
- En **sistemas distribuidos**, la **red** podría convertirse en un **cuello de botella** si no tiene la suficiente capacidad para el tráfico que genera el intercambio de mensajes entre procesos.

ESTADOS DE EJECUCIÓN DE UN PROCESO

Desde que se crea inicialmente un proceso, puede pasar por **diversos estados** y cambiar de uno a otro.

El **sistema operativo** gestiona los procesos y realiza los cambios de estado, teniendo en cuenta distintos **eventos** que pueden suceder durante el ciclo de vida del proceso:



ESTADOS DE EJECUCIÓN DE UN PROCESO

Los **cambios de estado** de los procesos están **controlados** por el **sistema operativo**, con el objetivo de conseguir dos cosas:

- Aprovechar los recursos del sistema, en particular del procesador o procesadores.
- Una ejecución lo más eficiente posible de todos los procesos.

Esto se consigue con una planificación en varios niveles:

1. **Planificador a largo plazo o de trabajos** (*job scheduler*): Un proceso debe estar en la memoria principal para poderse ejecutar.

   El planificador a largo plazo **decide** qué procesos son **admitidos** para su **ejecución**. Estos se cargan en la memoria principal y pasan a estado **listo** (*ready*).

   En **sistemas operativos interactivos** (los que manejan la mayoría de usuarios), no hay planificación a largo plazo, sino que son los propios **usuarios** quienes **lanzan** las **aplicaciones**, y esas se cargan **inmediatamente** en la **memoria**, y pasan a estado **listo**.

   En cambio, es **importante** en sistemas donde hay muchos **procesos no interactivos** y es el propio sistema operativo el que gestiona su lanzamiento.

ESTADOS DE EJECUCIÓN DE UN PROCESO

2. **Planificador a corto plazo o de procesador** (*CPU scheduler*): Se realiza para procesos cargados en la memoria principal.

   Su objetivo es **repartir** el **tiempo de procesador entre** todos los **procesos**, de manera que se consiga un **máximo aprovechamiento** del **procesador**, y que los **procesos** se **ejecuten** de la **manera más eficiente** posible.

   Un **proceso** en estado **listo** puede pasar a estado en **ejecución** y empezar a ejecutarse en el procesador.

   Un proceso en estado **ejecución** puede volver a estado **listo** para permitir que otro proceso se ejecute.

   Cuando un **proceso en ejecución** realiza una operación de **E/S,** pasa a estado **bloqueado**, y se pasa a **ejecutar** un **nuevo proceso** cuyo estado cambia **de listo a en ejecución**.

   Cuando la **operación** de **E/S** haya **concluido**, el proceso volverá a estado **listo**.

   El planificador a corto plazo funciona **basándose** en **interrupciones periódicas** que hacen que el procesador pase inmediatamente a ejecutar una **rutina de gestión de interrupción**.

   Esta **rutina** realiza la **planificación a corto plazo**, y **decide** si hay que **continuar** con la ejecución del **proceso actual**, o si hay que pasar a **ejecutar** un **nuevo proceso** y, en su caso, **cuál**.

ESTADOS DE EJECUCIÓN DE UN PROCESO

3. **Planificador a medio plazo**: Gestiona el **paso** de procesos de la **memoria principal** a la **secundaria** (**suspensión**) y viceversa (**reanudación**).

   Los **sistemas operativos actuales** funcionan, en general, con **memoria virtual**. Esta es una técnica que utiliza un espacio de direcciones virtual, **mayor** que el disponible en la **memoria física**, y utiliza para ello un medio de **almacenamiento secundario**.

   El **contenido** de cualquier dirección de la **memoria virtual** puede estar en la **memoria física** (memoria ***principal***) o en **almacenamiento secundario** (memoria ***secundaria***).

   Pero el **microprocesador sólo** puede acceder directamente a la **memoria principal**.

   Cuando **un proceso necesita más memoria** de la que hay disponible en ella, se consigue **pasando** parte de **sus contenidos a la memoria secundaria**.

   Cuando se **necesitan contenidos** que están en la **memoria secundaria**, deben **pasarse** antes a la **memoria principal**. Si **no** hay suficiente **espacio** en ella, antes hay que **pasar** algunos de sus **contenidos** a la memoria **secundaria**.

   Se produce así un **intercambio** o ***swap*** entre las memorias **principal y secundaria**.

   La **memoria virtual funciona** bien si los **intercambios** sólo se producen **ocasionalmente**. Si son **demasiado frecuentes**, se produce un trasiego constante entre las memorias principal y secundaria (*trashing*), y el **rendimiento** del sistema se **degrada** enormemente.

   Los **procesos** en la **memoria principal** que **no** se están **ejecutando** pueden **suspenderse**, y pasan a la **memoria secundaria** en los estados **suspendido listo** y **suspendido bloqueado**.

   Desde el estado **suspendido bloqueado** puede **pasar** al estado **suspendido listo** si se **completa** la operación **E/S**.

   También pueden **reanudarse** los procesos en estos estados para **pasar** a la **memoria principal**.

ESTADOS DE EJECUCIÓN DE UN PROCESO

Para la gestión de los procesos, el sistema operativo utiliza un **bloque de control de procesos** (**PCB**) para cada proceso.

El sistema operativo mantiene **colas de procesos** para cada uno de los estados.

También mantiene **colas** para cada **dispositivo de E/S**, con información de los procesos que tienen operaciones pendientes en dicho dispositivo.

La **utilización** de estas **colas** permite realizar, de manera muy eficiente, la **planificación** de **procesos**.

HILOS O HEBRAS (*THREADS*)

- También conocidos como **procesos ligeros** (*lightweight processes*).
- Un ***proceso en ejecución*** tiene inicialmente **un hilo**, pero se pueden crear más de manera sencilla y rápida.
- La **ejecución** de un **proceso termina** cuando **finaliza** la **ejecución** de **todos** sus **hilos**.
- Del mismo modo, cuando **termina** la **ejecución** de un **proceso**, se **acaba** la **ejecución** de **todos** sus **hilos**.

HILOS Y PROCESOS

Procesos

- Para **crear** un **proceso** es necesario **reservar** un espacio en **memoria**.
- La reserva de memoria **consume** una cantidad considerable de **tiempo** y **recursos**.
- La **comunicación** entre **procesos** requiere de ciertos **mecanismos especiales**.

Hilos o hebras (*threads*):

- La **creación** de un nuevo **hilo** para un proceso ya existente **no** requiere **reservar** e **inicializar espacio** en la **memoria**, porque este es **compartido** por todos los hilos de un mismo proceso.
- Si un **hilo** de un proceso **modifica** un **objeto** situado en la **memoria**, los **demás** hilos del mismo proceso pueden **ver** los **cambios** realizados.
- Se necesitan **mecanismos de sincronización** entre hilos para evitar problemas que puedan darse si los distintos hilos **modifican sin control objetos** situados en la **memoria compartida**.
- El **planificador** a **corto plazo** (*CPU scheduler*) gestiona de manera independiente los distintos hilos de un mismo proceso.

SERVICIOS (*SERVICES*)

Los **servicios** son un tipo particular de procesos que proporcionan determinados servicios a otros procesos.

Se ejecutan en **segundo plano** (*background*) y **no** son **directamente utilizados** por los **usuarios**.

Otros procesos, en cambio, se ejecutan en **primer plano** (*foreground*), y tienen una interfaz de usuario con la que interactúan los usuarios.

Los servicios **se ejecutan continuamente**. Normalmente, son **iniciados** por el **sistema operativo** durante su **arranque**.

Suelen proporcionar información acerca de su ejecución en ficheros de ***log***, en los que van anotando todos los **eventos** que se producen durante su **ejecución** y detalles acerca de las **acciones** que **realizan**.

Los servicios pueden **proporcionar servicios** a **otros procesos** en el **mismo ordenador**, y también **a otros procesos** en **otros ordenadores**. En este caso, suelen **comunicarse** a través de una **red de comunicaciones**, utilizando **protocolos de red estándares**, como **TCP** o **UDP**, de la familia de protocolos **TCP/IP**.

SERVICIOS (*SERVICES*)

Este tipo de servicios en red son un componente fundamental de los **sistemas distribuidos**.

En este contexto, las **aplicaciones servidoras**, o *servidores*, **prestan servicio a otras aplicaciones**, y las **aplicaciones clientes**, o *clientes*, **utilizan estos servicios**.

Una **aplicación** puede **proporcionar** un **servicio** a otras aplicaciones y **a la vez utilizar** los **servicios** que proporciona **otra aplicación**. Es decir, puede actuar **a la vez como cliente y servidor.**

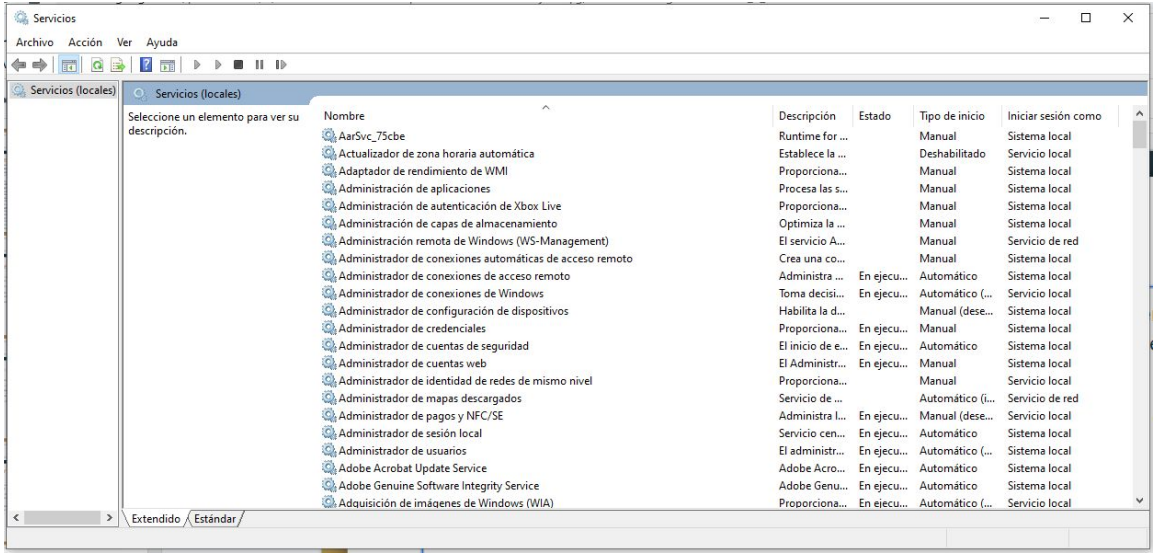
Una **técnica** frecuentemente **utilizada** por los **servicios** para **mejorar** el **tiempo de respuesta** ante peticiones de servicios de otros procesos es **crear un nuevo hilo** para **responder** a **cada nueva petición recibida**, o bien mantener un ***pool* de hilos**, de manera que **a cada nueva petición** se le puede **asignar** inmediatamente un **hilo** ya **disponible** en el ***pool***. De esta forma, el hilo principal queda libre para recibir nuevas peticiones. Un servidor de este tipo es un **servidor multihilo**.

SERVICIOS EN WINDOWS

En Windows, el **administrador de servicios** permite **arrancar y detener servicios**, así como consultar el estado de ejecución y obtener información adicional acerca de cada uno.

También se puede configurar el modo de arranque de cada servicio:

- **Automático**: para que arranque de forma automática cuando se inicia el sistema operativo.
- **Manual.**



SERVICIOS EN LINUX

En Linux existen sofisticados sistemas para gestión, control y configuración de servicios.

Con el siguiente comando se muestra una **lista con todos los servicios existentes** en el sistema. Aquellos para los que se muestra **`enabled`** en la columna **`STATE`** se arrancan automáticamente cuando arranca el sistema:

```
$ systemctl list-unit-files
```

Con el siguiente comando se muestran todos los **servicios cargados en el sistema**. Que estén cargados no significa que estén en ejecución actualmente. El estado de ejecución aparece en la columna **`SUB`**:

```
$ systemctl --type=service
```

Para cada servicio, se pueden ejecutar comandos del tipo:

```
$ systemctl acción servicio
```

Donde ***`servicio`*** es el nombre del servicio, y ***`acción`***, la acción que se quiere realizar sobre el servicio.