Programación Multiproceso

En general se dice que el número de **tareas** (programas) que puede realizar un ordenador en un instante de tiempo depende del número de procesadores que tiene. Si dispone de dos procesadores, será capaz de ejecutar **simultáneamente** dos tareas diferentes. Ésta fue la norma general durante décadas, y aunque el desarrollo tecnológico ha ido cambiado las cosas.

Al principio de la Informática, que un procesador sólo pudiese ejecutar una única tarea provocaba algo que hoy nos parece increíble: que el ordenador se quedase *bloqueado* ejecutando la tarea asignada. Es decir: arrancabas un programa y no podías hacer nada más hasta que terminase. Con el tiempo, se desarrolló toda una serie de tecnologías para sacar mejor provecho del procesador. La primera solución fue la capacidad de 'cortar' **varias tareas** en partes e ir **alternando** su ejecución en el procesador. Se seguía ejecutando una única tarea en cada instante de tiempo, pero ocurría tan deprisa que se *engañaba* al usuario haciéndole creer que se ejecutaban a la vez. Más adelante, se desarrollaron procesadores con varios **núcleos de ejecución**, que a efectos prácticos es *casi igual* que tener varios procesadores en uno. Por último, se desarrollaron tecnologías que mejoraban la **planificación** de tareas al máximo. Los procesadores de Intel llegan incluso a ejecutar código de un programa *antes* de que sea necesario, acertando en sus predicciones el 75% de las veces.

Lo que vamos a aprender en este tema tiene que ver con la capacidad de los procesadores de ejecutar varias tareas a la vez, lo que es un proceso, sus estados y la capacidad de comunicación entre ellos.

Introducción a la Multitarea

La **Multitarea** puede definirse como la capacidad de un ordenador de ejecutar varias tareas simultáneamente, independientemente del número de procesadores que tenga a su disposición. Existen dos tipos de multitarea:

- La **Real**, que sólo es posible cuando existen varios procesadores (o núcleos).
- La **Aparente**, en la que el Sistema Operativo alterna la ejecución de las tareas.

El procesador

El procesador es generalmente llamado el 'cerebro' del ordenador. Su función es **ejecutar** las instrucciones de un programa (tarea). Un **núcleo** es una unidad con capacidad de ejecución dentro de un procesador. Si un procesador tiene cuatro núcleos, se pueden ejecutar cuatro instrucciones a la vez, que podrían ser de diferentes tareas (o no).

Siguiendo esta lógica… ¿un procesador de un solo núcleo no puede hacer **Multitarea**? Pues sí que puede, pero no sería *Multitarea Real* sino *Multitarea Aparente*. El Sistema Operativo es capaz de alternar entre tareas – **Cambio de Contexto** – a una velocidad tal que parece que sí hace multitarea.

Programas, procesos, servicios

Denominamos **Proceso** a un programa en *ejecución*. Cuando queremos lanzar un programa, el Sistema Operativo toma el control y realiza toda una serie de pasos que, generalmente, son transparentes para el usuario. Simplemente vemos que el programa arranca y punto. En realidad, lo que sucede por detrás es bastante complicado. A modo de resumen:

- El código ejecutable del programa se carga en la memoria RAM del ordenador.
- Dedica un espacio para la pila del programa (memoria para variables etc.).
- Prepara el contador del programa (el que le dice la fila que está ejecutando).
- Prepara el puntero de pila (para las variables).
- Prepara la memoria estática.
- Prepara el **Bloque de Control de Procesos**.

El bloque de control del proceso (BCP) es donde el sistema operativo agrupa toda la información que necesita conocer respecto a un proceso particular. Habitualmente contiene información como:

- Identificativo de proceso (**PID**), un número que de forma única identifica al proceso
- Usuario que lo lanzó.
- Hora inicio.
- **Estado del proceso** (ver más abajo).
- …

Cuando todo está preparado, el Sistema Operativo **decide**, de entre todos los procesos disponibles, **cuál de ellos va a ejecutar**. Y como hemos mencionado antes, si tiene varios núcleos o procesadores, podrá ejecutar varios a la vez. La forma que tiene el Sistema Operativo de planificar la gestión de los procesos es la siguiente:

- Asigna cierto **tiempo de ejecución** a cada proceso. Cuando el tiempo acaba, cede el procesador o el núcleo a otro proceso.
- Asigna **prioridades** a los procesos. Un proceso prioritario tiene más probabilidades de recibir el procesador.

Cuando el tiempo de procesador para un proceso termina, se suspende temporalmente su ejecución hasta que vuelva a recibirlo. Para que el Sistema Operativo pueda volver a arrancarlo posteriormente, toda la información del proceso (BCP) se almacena en memoria.

En los Sistemas Operativos se distingue a demás entre Proceso y **Servicio**. Se define un Servicio como un programa que se ejecuta en segundo plano sin interacción con el usuario, aunque en realidad son también procesos (normalmente de baja prioridad).

Administradores de tareas

Todos los Sistemas Operativos modernos tienen herramientas que permiten al usuario informarse sobre los procesos que se están ejecutando en sus equipos. En el caso de Windows, a esta herramienta se le llama Administrador de Tareas, pero existen similares en todos los Sistemas Operativos.

- **Windows**: pulsas ctrl + alt + supr y accedes al Administrador de tareas.
- **MS-DOS**: tiene el comando **tasklist**.
- **Ubuntu**: en la interfaz gráfica, en Sistema/Administración/Monitor del sistema
- **Linux**: en modo consola, usas el comando **ps –AF**

Estados de un Proceso (Básico)

Generalmente se considera que un Proceso puede estar en uno de estos tres estados:

- **En ejecución**: El Proceso está actualmente usando un procesador o núcleo
- **Listo**: El proceso está detenido, pero únicamente necesita que el Sistema Operativo le ceda un procesador o núcleo para ejecutarse.
- **Bloqueado**: El proceso está detenido. No puede hacer absolutamente nada hasta que ocurra un evento externo que le permita dejar ese estado.

En un instante de tiempo concreto, un equipo puede tener un montón de Procesos en el estado Listo, otros tantos en estado Bloqueado, pero sólo unos pocos en Ejecución (tantos como procesadores o núcleos). El Sistema Operativo **decide** en cada momento qué Proceso debe de tener el procesador cambiando el estado de uno u otro Proceso. Es así como se consigue la *Multitarea Aparente*. Obviamente, esto ocurre tan rápido que es imposible que un humano no puede verlo.

Las Transiciones de un estado a otro también están predeterminadas.

1) **Bloqueo**: Normalmente sucede cuando el Proceso en ejecución necesita hacer algo que es extremadamente lento en comparación al procesador como, por ejemplo, leer un fichero del disco duro (operaciones E/S).

2) **Fin tiempo**: Generalmente, un Proceso pasa al estado Listo cuando se le agota el tiempo que el Sistema Operativo le ha dado para ejecutarse. El Proceso ni termina ni se destruye, sino que se queda en memoria esperando su oportunidad.
3) **Dispatch**: El Sistema Operativo ha decidido darle tiempo al Proceso para que se ejecute. El Proceso ahora dispone de tiempo de procesador pare ello.
4) **Desbloqueo**: Algo ha sucedido para sacar al Proceso bloqueado de su estado, y ahora puede competir por tiempo de procesador. Lo habitual es que una operación lenta de E/S como leer un fichero haya terminado, y por tanto el Proceso puede continuar.

En términos generales es **imposible** saber:

- En qué momento un proceso va a cambiar de estado.
- Cuánto tiempo va a estar en un estado.
- En qué orden se ejecutan los procesos.

Como ya se ha mencionado antes, los Sistemas Operativos tienen en cuenta tanto el **tiempo** como la **prioridad** de los Procesos para gestionar qué Proceso debería recibir el procesador. La forma exacta en que lo hace es bastante compleja, y de ello se encarga el **Scheduler** (Planificador). En líneas generales, podemos decir que **el usuario no puede intervenir directamente** en la gestión de procesos. No obstante, dispone de herramientas y comandos que le otorgan ciertas funcionalidades. También es posible en es controlar los bloqueos y los eventos de los procesos (hasta cierto punto) mediante la programación. Lenguajes de programación como **C/C++** que permiten a los programas lanzar procesos nuevos de forma sencilla.

Cambios de Estados de un Proceso

Partiendo de lo visto anteriormente sobre los Estados de un Procesos (Básico), podemos entender cómo hace un Sistema Operativo para gestionar Procesos. Vamos a partir de que tenemos un equipo en el que están corriendo tres procesos: **A**, **B** y **Nulo**. El Proceso Nulo es un proceso de *mínima prioridad* que sólo recibe tiempo de procesador cuando no hay ningún otro Proceso disponible. Windows dispone de uno; no así las distribuciones de Linux. En este momento, el proceso en **Ejecución** es A, mientras que B y Nulo están en estado **Listo**.

El Estado de los Procesos actualmente es el siguiente:

Acción	Ejecución	Listo	Bloqueado	Terminado
	A	B, Nulo		

Supongamos ahora que el Proceso A tiene que hacer una operación de Lectura de Disco Duro. Al tratarse de una operación infinitamente más lenta que el procesador, el Sistema Operativo decide Bloquearlo. No va a prestarle atención hasta que la operación termine.

Como el procesador está libre, decide coger un Proceso en espera para ejecutarlo: Dispatch. El más adecuado es B, dado que tiene más **Prioridad** que Nulo.

Por tanto, los Estados de los Procesos quedan así:

Acción	Ejecución	Listo	Bloqueado	Terminado
Lectura de Disco (A)	A	Nulo	B	

Supongamos ahora que el Proceso B requiere que alguien escriba por teclado. Se trata de otra operación lenta de Entrada/Salida, por lo que lo Bloquea. Escoge entre los Procesos en estado Listo el más adecuado, pero sólo hay uno: Nulo.

Los Estados de los Procesos quedan así:

Acción	Ejecución	Listo	Bloqueado	Terminado
Lectura Teclado (B)	Nulo		A, B	

Mientras las operaciones que colocaron A y B en el estado Bloqueo no terminen, no pueden competir por el procesador ni cambiar a estado Ejecución. En principio, cada ciclo de reloj del procesador debería de Finalizar el Tiempo del proceso, por lo que Nulo debería de irse a estado Listo y buscarse el Proceso más adecuado para pasar a Ejecución (Dispatch)… pero sólo tenemos a Nulo.

Varios ciclos de reloj más tarde, le llega la señal al Proceso A de que ya ha terminado la Lectura de Disco. Al hacerlo, el Proceso A se Desbloquea y pasa al estado Listo. Al hacerlo, ya puede competir por el Procesador, pero por el momento no es elegible, por tanto:

Acción	Ejecución	Listo	Bloqueado	Terminado
Señal de Fin de Lectura (A)	Nulo	A	B	

Llega una nueva señal al Proceso B indicando que se ha terminado de Leer de Teclado. Igual que antes, el Proceso B se Desbloquea y pasa a Listo. También Finaliza el Tiempo asignado a Nulo y pasa al estado Listo. La diferencia es que ahora se puede escoger entre A o Nulo para pasar a Ejecución (B no es elegible), así que:

Acción	Ejecución	Listo	Bloqueado	Terminado
Señal de Fin de Lectura (B)	A	Nulo, B		

Se ejecuta A con normalidad, pero ahora le llega la siguiente instrucción: **kill A**. Esto provoca que el proceso se destruya, por tanto, lo eliminamos de la planificación y pasamos al siguiente candidato a estado Ejecución:

Acción	Ejecución	Listo	Bloqueado	Terminado
Kill A	B	Nulo		A

Finalmente, el usuario decide terminar con B. En los Sistemas Operativos en los que existe Nulo, éste siempre va a existir, aunque se le destruya, incluso aunque no haya más Procesos que ejecutar. Por tanto:

Acción	Ejecución	Listo	Bloqueado	Terminado
Kill B	Nulo			A, B

Nótese que esta explicación es una **simplificación** de una Gestión de Procesos real. A modo de resumen:

Acción	Ejecución	Listo	Bloqueado	Terminado
	A	B, Nulo		
Lectura de Disco (A)	A	Nulo	B	
Lectura Teclado (B)	Nulo		A, B	
Señal de Fin de Lectura (A)	Nulo	A	B	
Señal de Fin de Lectura (B)	A	Nulo, B		
Kill A	B	Nulo		A
Kill B	Nulo			A, B

Ejercicio 1

Tenemos los siguientes procesos:

Acción	Ejecución	Listo	Bloqueado	Terminado
	A	B, Nulo	C, D	

Completa las siguientes tablas teniendo en cuenta que el **Scheduler** elegirá siempre el proceso con mayor prioridad. Ten en cuenta que:

- Al realizar la operación **fork** se crea un nuevo Proceso E
- La Prioridad de los Procesos es: D > E > B > A > C > Nulo

Acción	Ejecución	Listo	Bloqueado	Terminado
	A	B, Nulo	C, D	
Fork				
Señal de Fin de Lectura (D)				
Kill				
Señal de Fin de Lectura (C)				