* 1. **PROGRAMAS Y PROCESOS**

Programa: Conjunto de instrucciones que se pueden ejecutar directamente en una máquina física o virtual. Objeto estático almacenado en un fichero binario, y este en un disco duro.

Proceso: Instancia de un programa en ejecución. Entidad dinámica.

Ejecución de un programa: Comienza con la creación y ejecución de un proceso. Un proceso puede crear nuevos procesos sobre la marcha, pudiéndose crear una jerarquía de procesos (Linux: pstree, top (uso memoria y CPU); Windows: Process Explorer). Durante su ejecución se usan varios recursos del sistema.

* Memoria principal. Secuencia de celdas de memoria con la misma longitud en bits y cada una identificada por su posición. 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 que se crea una vez cargado el programa en la memoria. Guarda, en un registro especial (contador de programa, PC, program counter), la dirección en la memoria de la instrucción que se está ejecutando. Un proceso no puede operar directamente con los contenidos de la memoria. Estos antes deben traerse de ella y cargarse en registros del procesador. Las operaciones se realizan en una unidad aritmético-lógica o UAL (arithmetic and logical unit, ALU). Su resultado se obtiene en un registro desde el que se puede transferir 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 le ha otorgado acceso a un dispositivo E/S, y del estado de las operaciones realizadas sobre él.

Toda información asociada a un proceso se guarda en un bloque de control de proceso (PCB, process control block). Si el mismo programa se ejecuta varias veces, se creará cada vez un nuevo proceso con su propio PCB.

* 1. **MULTITAREA**

Los sistemas operativos actuales permiten ejecutar múltiples procesos a la vez, aunque exista solo un procesador. 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.

En cada momento hay un único proceso en ejecución, pero la rápida alternancia entre ellos hace que todos puedan seguir avanzando a lo largo del tiempo.

Antes de empezar a ejecutar un nuevo proceso, se guarda el estado de ejecución del proceso actual, para poder más adelante retomar su ejecución en el mismo punto en que se interrumpió. Se guardan los contenidos del contador de programa y de los registros del procesador, y se cargan los del nuevo proceso. Esto se conoce como cambio de contexto.

El rendimiento de un sistema aumenta si dispone de más de un procesador (sistema multiprocesador). En este caso, se pueden ejecutar en paralelo varios procesos, uno en cada procesador. Pero, normalmente, siempre habrá muchos más procesos que procesadores, por lo que, también en el caso de los sistemas multiprocesadores, se hará multitarea en cada procesador.

* 1. **PROCESOS Y SISTEMAS MONOPROCESADORES Y MULTIPROCESADORES**

Procesos concurrentes: Los que se ejecutan simultáneamente durante un intervalo dado de tiempo, ya sea de forma real (en sistemas multiprocesadores) o simulada (en sistemas monoprocesadores, o en un mismo procesador de un sistema multiprocesador).

Programación concurrente: Se ejecutan varios procesos concurrentes en un sistema. También hace referencia a las 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.

* + 1. **Sistemas monoprocesadores**

Multiprogramación: Ejecución concurrente de varios procesos en un sistema monoprocesador.

La técnica del cambio de contexto permite una multitarea efectiva y un mejor aprovechamiento del procesador, que se asigna a otro proceso durante el tiempo en que un proceso está inactivo, o a la espera de que se termine de realizar una operación de E/S

* + 1. **Sistemas multiprocesadores**

Se pueden clasificar, dependiendo de su arquitectura, en sistemas multiprocesadores fuertemente acoplados y sistemas multiprocesadores débilmente acoplados.

1. *Sistemas fuertemente acoplados*

Programación paralela: Ejecución de varios procesos concurrentes en un sistema multiprocesador. Se usa también este término para hacer referencia a las técnicas que permiten desarrollar programas que utilizan varios procesos concurrentes que pueden ejecutarse en distintos procesadores de un sistema multiprocesador.

Existe una memoria compartida para todos los procesadores, a la que todos ellos acceden a través de un bus de conexión. También acceden al sistema de E/S a través mismo bus.

Dos tipos:

1. Sistemas multiprocesadores simétricos (SMP): Los procesadores del sistema funcionan todos de igual manera.
2. Sistemas multiprocesadores asimétricos: Uno de los procesadores del sistema (master) controla al resto (slave). Este tipo de sistemas también se denominan master-slave.
3. *Sistemas débilmente acoplados*

Programación distribuida: Ejecución de varios procesos concurrentes en un sistema distribuido. Se usa también este término para hacer referencia al desarrollo de programas que utilizan varios procesos concurrentes que pueden ejecutarse en distintos procesadores de un sistema distribuido. En un sistema distribuido no hay memoria compartida ni red de conexión especifica entre los distintos ordenadores. La comunicación entre procesos se realiza mediante mensajes a través de la red de comunicaciones que comunica los distintos ordenadores (TCP o UDP).

No hay memoria compartida, cada procesador tiene su propia memoria y su propio sistema de E/S. Existe una red de comunicaciones entre los procesadores.

Una posibilidad es que los procesadores estén físicamente cercanos unos a otros y comunicados mediante una red a medida, de alta velocidad y con una estructura regular. Una alternativa son los clusters, que consisten en ordenadores convencionales conectados a través de redes locales. Un ejemplo son los clusters Beowulf, donde un ordenador actúa como controlador y servidor de archivos, permitiendo acceso a los nodos que generalmente no tienen teclado ni monitor. Estos nodos operan con sistemas UNIX o Linux, y añadir nuevos nodos solo requiere conectarlos a la red.

Otra posibilidad, muy común hoy en día, es que los procesadores estén en ordenadores independientes comunicados mediante una red de área local (LAN) o incluso de área amplia (WAN) a través de internet. Estos son los sistemas distribuidos. Se caracterizan por la heterogeneidad del hardware sobre el que están construidos y por la relativa independencia entre los sistemas que los constituyen.

Principales ventajas: Altamente escalables y configurables.

Principales inconvenientes: Sistemas heterogéneos y complejos de mantener. La sincronización entre procesos puede ser complicada. La comunicación entre procesos se realiza a través de la red con velocidad limitada, por lo que puede convertirse en un cuello de botella.

* 1. **VENTAJAS E INCONVENIENTES DE LA PROGRAMACIÓN CONCURRENTE**

La programación concurrente mejora el rendimiento del sistema al permitir que los procesadores ejecuten otros procesos mientras esperan operaciones de E/S, lo que también reduce el tiempo de respuesta al distribuir el tiempo de procesador entre múltiples procesos. Lo ideal es que las aplicaciones deben dividirse en procesos independientes que operen en paralelo con mínima interacción.

Un sistema multiprocesador debe ser flexible para distribuir dinámicamente la carga de trabajo y escalable para aumentar su capacidad con nuevos procesadores. Además, los sistemas multiprocesadores redundantes cuentan con procesadores suplementarios para garantizar alta disponibilidad y tolerancia a fallos, manteniendo la operatividad si uno de los procesadores falla.

Los principales inconvenientes vienen especialmente de la dificultad de implementar mecanismos adecuados de sincronización y comunicación entre procesos. 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.

* 1. **KERNEL (O NÚCLEO) DEL SISTEMA OPERATIVO Y LLAMADAS AL SISTEMA**

Kernel: Parte central, pequeña y muy optimizada del sistema operativo que da respuesta a multitud de eventos mediante un mecanismo de gestión de interrupciones, que pueden producirse por diversas causas:

* Eventos generados por el hardware.
* Llamadas al sistema. El kernel proporciona acceso para las aplicaciones a determinados servicios o funcionalidades del sistema operativo mediante llamadas al sistema. Suelen estar escritas en un lenguaje de bajo nivel, y suelen ser llamadas a través de APIs de alto nivel (POSIX, Win32, Win64).
* Interrupciones periódicas que invocan CPU scheduler. Este decide si se sigue ejecutando el proceso actual o se hace un cambio de contexto para pasar a ejecutar otro proceso.

Cuando sucede una interrupción, el procesador deja de ejecutar el proceso en curso y pasa a ejecutar una rutina de tratamiento de interrupción. Cuando esta finaliza, se reanuda la ejecución en el mismo lugar en que estaba antes de la interrupción. El procesador tiene dos modos de funcionamiento: modo kernel (supervisor) y modo usuario. Las rutinas de tratamiento de interrupciones se ejecutan en modo kernel.

* 1. **ESTADOS DE EJECUCIÓN DE UN PROCESO**

Desde que se crea inicialmente un proceso, puede pasar por diversos estados y cambiar de uno a otro. Los cambios de estado de los procesos están controlados por el sistema operativo, con el objetivo de conseguir un máximo aprovechamiento de los recursos del sistema, y en particular del procesador o procesadores, además de una ejecución lo más eficiente posible de todos los procesos. Esto se consigue con una planificación a varios niveles:

1. **Planificador a largo plazo o de trabajos (job scheduler).** Para que un proceso se ejecute, debe estar en la memoria principal. El planificador a largo plazo decide qué procesos se admiten y los carga en memoria, llevándolos al estado listo (ready). En sistemas operativos de escritorio, no hay una verdadera planificación a largo plazo, ya que los usuarios inician las aplicaciones, que se cargan inmediatamente en memoria. Sin embargo, la planificación a largo plazo es crucial en sistemas con muchos procesos no interactivos, donde el sistema operativo gestiona el lanzamiento de los mismos.
2. **Planificador a corto plazo o de procesador (CPU scheduler).** Se realiza para procesos cargados en la memoria principal, asignando el tiempo del procesador para maximizar su uso y eficiencia. Los procesos pueden pasar de estado listo a ejecución cuando el procesador los selecciona, y volver a listo para ceder el procesador a otro proceso. 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 utiliza interrupciones periódicas para activar una rutina del kernel que decide si el procesador debe seguir ejecutando el proceso actual o cambiar a otro. Esta rutina, parte del núcleo del sistema operativo, se encarga de gestionar la planificación, seleccionando el proceso más adecuado para ejecutar a continuación.
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 usan memoria virtual, una técnica que permite manejar un espacio de direcciones mayor que la memoria física disponible, utilizando almacenamiento secundario. El microprocesador solo accede a la memoria principal, por lo que cuando un proceso necesita más espacio, parte de su contenido se transfiere a la memoria secundaria (swap). Si un contenido en la memoria secundaria se necesita de nuevo, debe ser movido a la memoria principal, liberando espacio si es necesario. Este intercambio funciona bien si no es frecuente, ya que las operaciones con la memoria secundaria son lentas. Un exceso de intercambios causa trashing (trasiego), degradando el rendimiento. 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 de E/S que provocó su paso a estado bloqueado. También pueden reanudarse los procesos en estos estados para pasar a la memoria principal.

El sistema operativo utiliza un bloque de control de procesos (PCB) para gestionar cada proceso. Mantiene colas de procesos para cada estado (listo, bloqueado, etc.) y también colas para los dispositivos de E/S, donde se almacena la información de los procesos que tienen operaciones pendientes. Estas colas facilitan una planificación eficiente de los procesos.

* 1. **HILOS Y PROCESOS**

Los hilos (threads) se llaman a veces 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. Y a la inversa, si se termina la ejecución de un proceso, se acaba también la ejecución de todos sus hilos.

La creación de un nuevo hilo en un proceso no requiere reservar memoria adicional, ya que todos los hilos de un proceso comparten el mismo espacio de memoria, lo que facilita su comunicación. Si un hilo modifica un objeto en memoria, los demás hilos pueden ver ese cambio. Para evitar conflictos por modificaciones simultáneas, se utilizan mecanismos de sincronización. El CPU scheduler gestiona cada hilo de un proceso de forma independiente, permitiendo una mejor distribución de los recursos del procesador.

* 1. **SERVICIOS**

Los servicios son procesos que se ejecutan en segundo plano (background) y proporcionan funciones a otros procesos, sin interactuar directamente con los usuarios. Se inician normalmente al arrancar el sistema operativo y funcionan de manera continua. Los servicios registran su actividad 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 atender tanto a procesos en el mismo ordenador como a procesos en otros ordenadores a través de redes de comunicaciones. Para esto, utilizan protocolos estándar como TCP o UDP, pertenecientes a la familia de protocolos TCP/IP. Estos servicios en red son fundamentales en sistemas distribuidos, donde las aplicaciones servidoras prestan servicios a otras aplicaciones, y las aplicaciones clientes consumen estos servicios. Una misma aplicación puede actuar simultáneamente como cliente y servidor, proporcionando servicios a otras aplicaciones mientras utiliza servicios ofrecidos por otras.

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.

* + 1. **Servicios en Windows**

El administrador de servicios permite arrancar, detener y consultar el estado de ejecución e información adicional. Se puede configurar el modo de arranque para cada servicio: automático o manual.

* + 1. **Servicios en Linux**

Con el siguiente comando, se muestra una lista con todos los servicios existentes en el sistema.

$ systemctl list-unit-files

Con el siguiente comando, se muestran todos los servicios cargados en el sistema.

$ systemctl --type=service

Para cada servicio, se pueden ejecutar comandos del tipo:

$ systemctl acción servicio

* 1. **CONCURRENCIA EN JAVA**

PDF de Jonathan (tema 3). También tengo pendiente hacer apuntes del tema 1 (procesos, RR, FIFO…)

* + 1. **Redirección de entrada y salida estándares y de error**

Las clases ProcessBuilder y Process proporcionan diversos mecanismos para redireccionar y gestionar la salida y entrada de los procesos. Esto podría no funcionar bien en algunas plataformas, debido a limitaciones en los buffers utilizados para ellas por el sistema operativo particular. Estos mecanismos se aprovechan en algunos de los programas de ejemplo que se presentan a continuación.

* Los conceptos de entrada y salida estándares y de salida de error se introdujeron en el sistema operativo UNIX, y posteriormente en Linux y Windows.
  + Entrada estándar o stdin (standard input). Es un stream de entrada del que puede leer datos. Normalmente, se leen datos sobre los que se realiza algún procesamiento. Por defecto, suele estar asociada al teclado, de manera que el programa puede leer todo el texto que se introduce por teclado.
  + Salida estándar o stout (standard output). Es un stream de salida en el que se pueden escribir datos. Una manera sencilla en que un programa puede generar su salida es escribirla en la salida estándar. Por defecto, suele estar asociada a la pantalla, de manera que todo el texto que se envía a la salida estándar se escribe en la pantalla.
  + Salida de error o stderr (standard output). Es un stream de salida en el que se pueden escribir datos. También suele estar asociada a la pantalla, de manera que todo el texto que se envía a la salida de error se escribe en la pantalla.

