Tema 1: Programación Multiproceso

1. Introducción: Aplicaciones, Ejecutables y Procesos.

Una aplicación es un tipo de programa informático, diseñado como herramienta para resolver de manera automática un problema específico del usuario.

Recordemos, que un programa es el conjunto de instrucciones que ejecutadas en un ordenador realizarán una tarea o ayudarán al usuario a realizarla.

Nosotros, como programadores y programadoras, creamos un programa, escribiendo su código fuente; con ayuda de un compilador, obtenemos su código binario o interpretado. Este código binario o interpretado, lo guardamos en un fichero. Este fichero, es un fichero ejecutable, llamado comúnmente: ejecutable o binario. Un ejecutable es un fichero que contiene el código binario o interpretado que será ejecutado en un ordenador.

Ahora, ¿qué es un proceso? De forma sencilla, un proceso, es un programa en ejecución. Pero, es más que eso, un proceso en el sistema operativo (SO), es una unidad de trabajo completa y el SO gestiona los distintos procesos que se encuentren en ejecución en el equipo.

Un proceso existe mientras que se esté ejecutando una aplicación. Es más, la ejecución de una aplicación, puede implicar que se arranquen varios procesos en nuestro equipo; y puede estar formada por varios ejecutables y librerías.

2. Ejecutables. Tipos.

Según el tipo de código que contenga un ejecutable, los podemos clasificar en:

- **Binarios:** Formados por un conjunto de instrucciones que directamente son ejecutadas por el procesador del ordenador. Este código se obtiene al compilar el código fuente de un programa y se guarda en un fichero ejecutable. Este código solo se ejecutará correctamente en equipos cuya plataforma sea compatible con aquella para la que ha sido compilado (no es multiplataforma).
- **Interpretados:** Código que suele tratarse como un ejecutable, pero no es código binario, sino otro tipo de código, que en Java, por ejemplo se llama bytecode. Está formado por códigos de operación que tomará el intérprete (en el caso de Java, el intérprete es la máquina virtual Java o JRE). Ese intérprete será el encargado de traducirlos al lenguaje máquina que ejecutará el procesador. El código interpretado es más susceptible de ser multiplataforma o independiente de la máquina física en la que se haya compilado.
- **Librerías:** Conjunto de funciones que permiten dar modularidad y reusabilidad a nuestros programas. Las hemos incluido en esta clasificación, porque su contenido es código ejecutable, aunque ese código sea ejecutado por todos los programas que invoquen las funciones que contienen.

3. Multiprogramación o multitarea.

Multiprogramación o multitarea es la técnica que permite que dos o más procesos ocupen la misma unidad de memoria principal y que sean ejecutados al "mismo tiempo".

1. Ventajas de la multiprogramación.

- Permite la existencia de varios procesos en ejecución.
- Permite el servicio interactivo simultáneo a varios usuarios de manera eficiente.
- Aprovecha los tiempos que los procesos pasan esperando a que se completen sus operaciones de E/S, por lo que aumenta el uso de la CPU.
- Las direcciones de los procesos son relativas, el programador no se preocupa por saber en dónde estará el proceso dado que el sistema operativo es el que se encarga de convertir la dirección lógica en física.

4. Multiprogramación y Multiproceso.

Multiprogramación es distinto a multiproceso. El multiproceso implica la existencia de varios procesadores en el mismo sistema, ejecutando por ejemplo simultáneamente diferentes trabajos, pudiendo ser usados en multiprogramación.

En cambio un sistema de multiprogramación no podría ser utilizado en multiproceso si solo tuviera un procesador.

Tipos de multitarea :

- **Multitarea Nula:** es aquel SO que carece de multitarea, por ejemplo MS-DOS.
- **Multitarea Corporativa, Cooperativa o No apropiativa:** es aquella donde los procesos de usuario son los que ceden la CPU al SO a intervalos regulares. Es el tipo de multitarea de los SO Windows anteriores a 1995 como Windows 3.11.
- **Multitarea Real:** es aquella en donde el SO ejecuta los procesos realmente al mismo tiempo, haciendo uso de múltiples procesadores. En el caso de que los procesos o tareas sean más que la cantidad de procesadores, éstos comienzan a ejecutarse como en la multitarea preferente. Es el tipo de multitarea de los SO modernos. En este caso tenemos un sistema multiproceso.

5. Gestión de procesos.

Tipos de procesos que se ejecutan en el sistema:

- **Por lotes:** Están formados por una serie de tareas, de las que el usuario solo está interesado en el resultado final. El usuario, solo introduce las tareas y los datos iniciales, deja que se realice todo el proceso y luego recoge los resultados.
- **Interactivos:** Aquellas tareas en las que el proceso interactúa continuamente con el usuario y actúa de acuerdo a las acciones que éste realiza, o a los datos que suministra.

- **Tiempo real:** Tareas en las que es crítico el tiempo de respuesta del sistema. Por ejemplo: el ordenador de a bordo de un automóvil.

1. *Herramientas gráficas para la gestión de procesos.*

Tanto los sistemas Windows como GNU/Linux proporcionan herramientas gráficas para la gestión de procesos. En el caso de Windows, se trata del Administrador de tareas, y en GNU/Linux del Monitor del sistema. Funcionalidades:

- Listado de todos los procesos que se encuentran activos en el sistema, mostrando su PID, usuario y ubicación de su fichero ejecutable.
- Posibilidad de finalizar procesos.
- Información sobre el uso de CPU, memoria principal y virtual, red...
- Posibilidad de cambiar la prioridad de ejecución de los procesos.

2. *Estados de un Proceso.*

En el siguiente gráfico, podemos ver un esquema muy simple de cómo podemos planificar la ejecución de varios procesos en una CPU:

Podemos quedarnos con los siguientes estados en el ciclo de vida de un proceso:

- **Nuevo:** Proceso nuevo, creado.
- **Listo:** Proceso que está esperando la CPU para ejecutar sus instrucciones.
- **En ejecución:** Proceso que actualmente, está en turno de ejecución en la CPU.
- **Bloqueado:** Proceso que está a la espera de que finalice una E/S.
- **Suspendido:** Proceso que se ha llevado a la memoria virtual para liberar, un poco, la RAM del sistema.
- **Terminado:** Proceso que ha finalizado y ya no necesitará más la CPU.

3. *Planificación de procesos por el Sistema Operativo.*

Dentro de la gestión de procesos vamos a destacar dos componentes del SO que llevan a cabo toda la tarea: el cargador y el planificador.

El cargador es el encargado de crear los procesos. Cuando se inicia un proceso (para cada proceso), el cargador, realiza las siguientes tareas:

- Carga el proceso en memoria principal.
- Crea una estructura de información llamada PCB (Bloque de Control de Proceso). Entre otros datos, el PCB estará formado por:
  - Identificador del proceso o PID.
  - Estado actual del proceso.
  - Espacio de direcciones de memoria donde comienza la zona de memoria reservada al proceso y su tamaño.
  - Información para la planificación: prioridad, quamtum, estadísticas…
  - Información para el cambio de contexto: valor de los registros de la CPU, entre ellos el contador de programa y el puntero a pila.

El planificador se encarga de decidir qué proceso se ejecuta y cuánto tiempo se ejecuta. Los algoritmos de planificación más importantes son:

- **Round-Robin:** Es aquél en el que cada proceso puede ejecutar sus instrucciones en la CPU durante un quamtum. Si no le ha dado tiempo a finalizar en ese quamtum, se coloca al final de la cola de procesos listos, y espera a que vuelva su turno de procesamiento.
- **Por prioridad:** En este algoritmo, se asignan prioridades a los distintos procesos y la ejecución de estos se hace de acuerdo a esa prioridad asignada.
- **Múltiples colas:** Es una combinación de los dos anteriores y el implementado en los sistemas operativos actuales. Todos los procesos de una misma prioridad, estarán en la misma cola. Cada cola será gestionada con el algoritmo Round-Robin. Los procesos de colas de inferior prioridad no pueden ejecutarse hasta que no se hayan vaciado las colas de procesos de mayor prioridad.

En la planificación (scheduling) de procesos se busca conciliar los siguientes objetivos:

- **Equidad:** Todos los procesos deben poder ejecutarse.
- **Eficacia:** Mantener ocupada la CPU un 100 % del tiempo.
- **Tiempo de respuesta:** Minimizar el tiempo de respuesta al usuario.
- **Tiempo de regreso:** Minimizar el tiempo que deben esperar los usuarios de procesos por lotes para obtener sus resultados.
- **Rendimiento:** Maximizar el número de tareas procesadas por hora.

4. Cambio de contexto en la CPU.

Una CPU tiene unas pequeños espacios de memoria(llamados registros), en los que se almacenan temporalmente la información que, en cada instante, necesita la instrucción que esté procesando la CPU. El conjunto de registros de la CPU es su estado.

Entre los registros, destacamos el Registro Contador de Programa y el puntero a la pila.

- El Contador de Programa en cada instante almacena la dirección de la siguiente instrucción a ejecutar. Este Contador de Programa nos permitirá continuar en cada proceso por la instrucción en dónde lo hubiéramos dejado todo.
- El Puntero a Pila en cada instante apunta a la parte superior de la pila del proceso en ejecución. En la pila de cada proceso es donde será almacenado el contexto de la CPU y de donde se recuperará cuando ese proceso vuelva a ejecutarse.

5. Servicios. Hilos.

Un proceso, estará formado por, al menos, un hilo de ejecución. Un proceso es una unidad pesada de ejecución. Si el proceso tiene varios hilos, cada hilo, es una unidad de ejecución ligera.

Un servicio es un proceso que, normalmente, es cargado durante el arranque del sistema operativo. Recibe el nombre de servicio, ya que es un proceso que queda a la espera de que otro le pida que realice una tarea.

6. Comandos para la gestión de procesos.

Los comandos que nos interesa conocer para la gestión de procesos son:

- **Windows:**
  - *tasklist:* Lista los procesos presentes en el sistema. Mostrará el nombre del ejecutable; su correspondiente Identificador de proceso y el porcentaje de uso de memoria entre otros datos.
  - *taskkill:* Mata procesos. Con la opción /PID especificaremos el Identificador del proceso que queremos matar.
- **GNU/Linux:**
  - *ps:* Lista los procesos presentes en el sistema. Con la opción "aux" muestra todos los procesos del sistema independientemente del usuario que los haya lanzado.
  - *pstree:* Muestra un listado de procesos en forma de árbol, mostrando qué procesos han creado otros. Con la opción "AGu" construirá el árbol utilizando líneas guía y mostrará 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" ejecutará el comando con una prioridad 5. Por defecto la prioridad es 0. Las prioridades están entre -20 (más alta) y 19 (más baja).