TEMA 2: FUNCIONES DEL SISTEMA OPERATIVO INFORMÁTICO

2.1 CONCEPTOS BÁSICOS

2.1.1 Los procesos

No hay que confundir procesos con archivos o programas. Un **proceso** es un programa en ejecución por parte del usuario o del sistema (en el caso de que lo ejecute el sistema entonces se denominan **servicios**).

Cada proceso se compone de un código que se ejecuta (**programa**) y unas estructuras de datos, estando ambos cargados en memoria. Se puede definir un proceso como una **instancia** de un programa en ejecución, de manera que el programa lleva los datos asociados a cada ejecución del mismo. Para ver cómo funciona esto, se puede abrir dos veces la calculadora de Windows (dos instancias del programa calc.exe); los procesos son diferentes y tienen asociados datos diferentes y en sus diferentes ejecuciones.

La estructura de datos asociada al proceso sirve para identificar unívocamente cada proceso y controlar todos los aspectos de su ejecución. Cada estructura contiene básicamente los siguientes datos:

- **Código máquina del programa**.
- **Datos del programa** (variables y parámetros del programa).
- **Pilas**. Cada proceso tiene asociadas una o varias pilas para almacenar direcciones de retorno a subrutinas u otros datos.
- **Bloque de Control del Proceso** (PCB, Process Control Black), conteniendo:
    - **Estado actual del proceso**: inactivo, preparado, en ejecución y suspendido.
    - **Identificación unívoca del proceso**. Al proceso se le asigna un PID (código identificador de proceso).
    - **Identificador del usuario propietario del proceso**.
    - **Prioridad del proceso**. Cada proceso tiene asignada una prioridad, de forma que, en cualquier instante, el proceso que mayor prioridad tiene asignada, de entre los que están en estado ESPERANDO, es el que se ejecutará.
    - **Zona de memoria asignada**. Cada proceso debe tener una zona de memoria independiente que no puede ser interferida por otros procesos.
    - **Recursos asociados** al proceso, como ficheros, semáforos, etc.

Las **hebras** (threads o **hilos**), implementadas en algunos sistemas operativos, reducen la recarga que supone la comunicación entre procesos cooperantes y la conmutación entre ellos. Un proceso puede estar compuesto por varias hebras o hilos de ejecución, cada una de las cuales puede ser considerada como un proceso ligero con un estado reducido. Las hebras de un proceso se caracterizan por:

- Pueden ejecutarse de forma concurrente, acelerando su ejecución.
- Las hebras son la unidad de ejecución más pequeña a efectos de planificación.
- El proceso es la unidad de ejecución propietaria de recursos, que son compartidos por las hebras pertenecientes al proceso, de manera que la comunicación entre estas hebras es más fácil y no se requiere el cambio de contexto del proceso para pasar de una hebra a otra, pero la protección de los recursos utilizando hebras de un mismo proceso no es controlada por el sistema operativo (así, por ejemplo, una hebra puede leer/escribir la memoria utilizada por otra perteneciente al mismo proceso).

En definitiva, las hebras proporcionan beneficios de velocidad y compartición, sacrificando la protección por el rendimiento. Fueron introducidas inicialmente en el sistema operativo UNIX. En los sistemas operativos que no implementan hebras, se pueden considerar todos los procesos formados por una única hebra, que seguiría siendo la mínima unidad de ejecución planificable.

2.1.1.1 MONOPROGRAMACIÓN Y MULTIPROGRAMACIÓN

En los sistemas operativos monoproceso primitivos se ubicaba únicamente un programa en la memoria principal, junto con el código del sistema operativo. Hasta que no terminaba su ejecución, no se cargaba el siguiente programa.

La **monoprogramación** no es eficiente, debido a que la memoria se suele ocupar parcialmente y el tiempo del procesador es pobremente aprovechado, ya que la CPU se ve obligada a esperar cuando el único programa en ejecución realiza operaciones de E/S sobre periféricos lentos.

La **multiprogramación** permite aprovechar de forma más eficiente los recursos del sistema, en concreto el tiempo de CPU, que es el recurso de mayor coste. Mientras unos procesos realizan operaciones de E/S, la CPU es asignada a otros procesos activos mejorando el rendimiento del sistema.

Si el sistema es multitarea, es común que existan varios procesos que se ejecutan en un mismo intervalo de tiempo. Cada proceso tiene asociado un identificador numérico llamado PID (Process Identificator o Identificador de Proceso) y está asociado con el identificador del proceso que lo inició o PPID (Parent Process Identificator o Identificador de Proceso Padre).

2.1.2 Los archivos

El sistema operativo proporciona una visión uniforme para todos los sistemas de almacenamiento, definiendo una unidad lógica de almacenamiento denominada **archivo**, que es un conjunto de información relacionada definida por su creador.

Los archivos siempre contienen información binaria, pero según su tipo y formato, la interpretación será diferente: líneas de código o instrucciones (programa), caracteres (archivos de texto), píxeles (formatos gráficos), notas musicales (archivos MIDI), registros (archivos de base de datos), etc.

Los archivos son referenciados por su nombre. Además, tienen otras propiedades o **atributos**, como son su tipo, fecha y hora de creación, nombre o identificador del creador, tamaño, etc.

Los archivos almacenan la información de forma permanente (no volátil), trascendiendo a la duración de los procesos que los utilizan o generan.

Las condiciones esenciales para el almacenamiento de la información a largo plazo son:

- Debe ser posible almacenar una cantidad muy grande de información.
- La información debe sobrevivir a la conclusión del proceso que la utiliza.
- Debe ser posible que varios procesos tengan acceso concurrente a la información.

Los directorios son tablas simbólicas de archivos. Una entrada típica puede contener:

- Nombre.
- Puntero de acceso al archivo, dirección de comienzo en el disco.
- Lista de atributos.

Las situaciones que se pueden dar son:

- **Directorio de nivel único**. Es un sistema usado en los primeros microcomputadores. En realidad, no contempla la posibilidad de directorios; únicamente reserva un espacio limitado en el disco para almacenar las entradas de ficheros (directorio raíz).
- **Árbol de directorios**. Las entradas del directorio correspondiente tienen un atributo más que indica si corresponde a un archivo (información en sí misma) o a un subdirectorio (contenedor de entrada de archivos).

2.1.3 Las llamadas al sistema

Las **llamadas al sistema** es el conjunto de instrucciones que sirve como interfaz con el sistema operativo para solicitar los servicios que ofrece. Ejemplos:

- **Procesos**. Entre ellas se encuentran crear un proceso, cambiar prioridades, etc. Un proceso es creado por otro mediante la llamada correspondiente al sistema. Estos procesos mantienen una relación padre-hijo.
- **Memoria**. Entre ellas se encuentran solicitar memoria, liberar memoria, etc.
- **Archivos**. Entre ellas se encuentran crear, renombrar, leer, escribir, etc.
- **Protección**. Entre ellas se encuentran propietarios, permisos, etc.

2.1.4 El núcleo del sistema operativo

El **Kernel**, o núcleo, es la parte del sistema operativo que interactúa directamente con el hardware. Cuando arranca el ordenador, se carga en memoria y permanece allí, realizando funciones básicas:

- **Comunicación y conmutación de procesos**. Lleva la cuenta de los procesos activos, trasladando el control de la CPU de un proceso a otro y almacenando el estado del sistema (contexto) en estructuras de datos. El planificador o asignador de recursos es el responsable de esta asignación de la CPU a cada uno de los procesos. La comunicación entre procesos se puede hacer mediante **semáforos** o mensajes.

  NOTA:

  Los semáforos son herramientas de sincronización de procesos, que permiten ejecutar un proceso que se detiene en espera de que otro (ejecutado concurrentemente) le pase un dato o le dé permiso.

- **Control de interrupciones**. Una interrupción provoca la detención momentánea de la ejecución del programa y la bifurcación a una posición de memoria donde comienza la rutina de tratamiento de la interrupción (RTI), que se encarga de realizar las operaciones de E/S y, posteriormente, retorna al punto donde se había interrumpido la ejecución del programa.
- **Manejo de condiciones de error**. En este apartado es preciso distinguir la siguiente terminología:
  - **Fallo** (fault). Es una imperfección en los componentes o en el diseño de un sistema que puede conducirlo a estados erróneos.
  - **Error** (error). Es cada una de las diferencias entre el estado erróneo y el estado válido del sistema. Se presentan como manifestación de un fallo en un componente del sistema, el cual deberá ser cambiado para llevar al sistema a un estado válido. Los errores pueden producirse a cierta distancia de los lugares donde se origina el fallo.

- **Avería** (failure). Ocurre cuando el servicio prestado difiere del servicio especificado. La situación de avería se produce en el sistema cuando los errores se acumulan y no son corregidos.

La manifestación de un fallo producirá errores en el estado interno del sistema, lo que finalmente puede conducir (o no) a una avería en él.

Los errores pueden ser corregidos en un determinado sistema informático, pero los fallos (defectos) causantes de esos errores permanecerán mientras no se cambie la configuración del sistema. En unos casos, será necesario modificar el diseño; en estas situaciones se producen errores aun cuando todos los componentes (excepto el diseño) se comporten de forma correcta. En otros casos, será necesaria la sustitución o reparación del componente o componentes que contienen fallos.

2.1.5 El intérprete de comandos

El **intérprete de comandos** proporciona una interfaz entre los usuarios interactivos y el sistema operativo. Desde este punto de vista, se pueden distinguir **sistemas operativos orientados a carácter**, que solo aceptan líneas de comandos sintácticamente correctas (Unix, MS-DOS) y **sistemas operativos gráficos**, que presentan facilidades para gestionar el sistema operativo a través de interfaces gráficas (Windows, Mac OS, Linux, ChromeOS,...)

La **línea de comandos**, también llamada **CLI** (Command Line Interface o Interfaz de Línea de Comandos) o línea de mandatos, es un método muy simple de interacción con un ordenador: el usuario escribe la operación que desea realizar (orden), pulsa la tecla [**Intro**], espera a que el ordenador la procese y finalmente muestre en pantalla los resultados obtenidos.

2.2 FUNCIONES DEL SISTEMA OPERATIVO

A continuación, se muestran las funciones principales que realiza todo sistema operativo:

- **Control de la ejecución de los programas**. Para ello, acepta los trabajos, administra la manera en que se realizan, les asigna los recursos y los conserva hasta su finalización.
- **Administración de periféricos**. Coordinando y manipulando los dispositivos conectados al ordenador.
- **Gestión de permisos y de usuarios**. Adjudica los permisos de acceso a los usuarios y evita que las acciones de uno afecten el trabajo que está realizando otro.
- **Control de concurrencia**. Establece prioridades cuando diferentes procesos solicitan el mismo recurso.
- **Control de errores**. Gestiona los errores de hardware y la pérdida de datos.
- **Administración de memoria**. Asigna memoria a los procesos y gestiona su uso.

- **Sistema de protección**. Debe proporcionar seguridad tanto para los usuarios como para el software y la información almacenada en los sistemas.
- **Sistema intérprete de comandos** (shell). Proporciona la interfaz interactiva entre los usuarios y el sistema operativo.
- **Sistema de comunicaciones**. El sistema de comunicaciones no es una función necesaria en un sistema operativo para un ordenador aislado, pero esta modalidad es una especie en peligro de extinción y cada vez es más difícil encontrar un ordenador no conectado a la red, aunque sea en ambientes domésticos. Por esta razón, todos los sistemas operativos actuales incluyen protocolos y servicios de red.

En concordancia con estas funciones principales, es posible analizar la estructura de un sistema operativo en cuatro niveles. Cada uno de los niveles se comunica con el inmediatamente inferior y superior coordinando sus funciones.

- **Nivel 1. Gestión del procesador**. En este nivel se encuentra la parte del sistema operativo encargada de la gestión de la CPU. En los sistemas operativos multiproceso (es decir, que pueden ejecutar varios procesos a la vez), este nivel se encarga de compartir la CPU entre los distintos procesos realizando funciones de sincronización, conmutación de la CPU y gestión de interrupciones.
- **Nivel 2. Gestión de dispositivos**. En este nivel se realiza la gestión de las entradas/salidas (E/S) en función de los dispositivos existentes. Entre otras se encarga de las funciones de creación de procesos de E/S, asignación y liberación de dispositivos E/S y planificación de la E/S.
- **Nivel 3. Gestión de memoria y procesos**. Este nivel es el encargado de repartir la memoria disponible entre los procesos, así como su creación, comunicación y destrucción. Se realizan funciones de asignación y liberación de memoria y el control de violación de acceso a zonas de memoria no permitidas.
- **Nivel 4. Gestión de la información**. El objetivo de este nivel es el de gestionar el espacio de nombres lógicos y la protección de la información realizando funciones de creación y destrucción de ficheros y directorios, apertura y cierre de ficheros, lectura y escritura de ficheros y protección de acceso.

Es importante destacar que un mismo sistema operativo puede trabajar en múltiples plataformas hardware, por lo que debe poder adaptarse a las peculiaridades de cada una de ellas.