3. Funciones

3.1. Objetivos de un sistema operativo

Los objetivos de un sistema operativo son, por un lado, incrementar la productividad de los usuarios y facilitar su uso del equipo. Por otro lado, tratan asimismo de ofrecernos un entorno cómodo que resulte más abstracto o independiente con respecto al hardware. También pretende optimizar la utilización del hardware para maximizar su rendimiento.

Figura 3.1 Relación entre el usuario y el hardware a través del Sistema Operativo como intermediario.

Otros propósitos importantes que conviene mencionar son: Gestionar los recursos del hardware y del software decidiendo quién, cuándo, cómo y durante cuánto tiempo se utilizará un recurso por parte de un proceso; resolver conflictos entre peticiones concurrentes y preservar la integridad del sistema.

3.2. Funciones

Las funciones más importantes de un sistema operativo son las siguientes:

- Administrar el procesador —o los procesadores, en caso de contar con sistemas multiprocesador—.
- Administrar la memoria.
- Comunicar los dispositivos con los usuarios de manera transparente.
- Organizar los datos, garantizando así un acceso rápido y seguro a los mismos.
- Gestionar las comunicaciones de red.
- Facilitar las entradas y salidas.
- Ofrecer técnicas de recuperación de errores.
- Evitar que unos usuarios interfieran en el trabajo de otros.
- Generar estadísticas y archivos de registro de eventos del sistema.
- Compartir el hardware y los datos entre múltiples usuarios.

Todas estas funciones se pueden resumir en dos funciones primordiales e indispensables:

- **Gestionar el hardware**, lo que hace referencia a que es el responsable de gestionar el hardware del sistema de la manera más eficiente posible.
- **Facilitar el trabajo al usuario**, pues permite y facilita al usuario la labor de comunicación con la máquina.

A continuación, nos centraremos en detallar algunas de estas funciones de manera más exhaustiva:

A. Gestión de procesos

En el punto anterior hemos señalado la diferencia entre un proceso y un programa, pero ahora veremos qué hace realmente el sistema operativo con los procesos y qué responsabilidad tiene sobre ellos.

Los procesos se pueden encontrar en diferentes estados.

Figura 3.2 Estados de un proceso.

Cuando un usuario decide iniciar un proceso, este llega al estado NUEVO. A continuación, tras ser es admitido, pasa al estado PREPARADO. Más adelante, el **planificador** de procesos escoge el proceso que pasará a estado EN EJECUCIÓN.

Cuando un proceso se encuentra en ejecución, pueden ocurrir 3 cosas:

1. Que el proceso acabe de manera normal, pasando así al estado FINALIZADO.
2. Que el proceso se vea detenido por una interrupción de sistema, por un proceso con mayor prioridad o porque, simplemente, se ha agotado su tiempo disponible (*quantum*), con lo que pasará de nuevo al estado PREPARADO a la espera de que le llegue, de nuevo, su turno.
3. Que el proceso necesite recibir una Entrada o realizar una Salida (por ejemplo, esperar la entrada de teclado del usuario, o la impresión de un documento). En este caso, el proceso pasa al estado SUSPENDIDO, que abandonará para pasar al estado PREPARADO tan pronto como la Entrada, la Salida o el evento que estaba esperando suceda.

El **planificador** puede escoger el siguiente proceso de acuerdo con diferentes criterios y algoritmos:

- FCFS, o *First Come First Served*: se atiende primero al proceso que llega antes.
- RR, o *Round Robin*: el procesador se asigna en turnos rotatorios a los procesos que están preparados durante un fragmento de tiempo pequeño llamado *quantum*.
- SJF, o *Shortest Job First*: se atiende primero el proceso más corto.

- SRTF, o *Shortest Remaining Time First*: se ejecuta primero el proceso al que le quede menos tiempo para terminar.

Todo este cambio de estados es responsabilidad del sistema operativo, que utiliza, para llevarlos a cabo, una tabla de control de procesos o *Process Control Block* (PCB). La información que contiene el PCB es, entre otros datos, el identificador del proceso y el identificador de su padre (es decir, de quien lo invoca), el identificador del usuario y del grupo al que pertenece el usuario que lanza el proceso, la información sobre el estado del proceso, el planificador, los segmentos de memoria y los recursos asignados.

Figura 3.3. PCB (Process Control Block).

B. Gestión de memoria

El gestor de la memoria es el responsable de:

- Asignar zonas de memoria a los procesos cuando la requieran y liberarla cuando hayan acabado.
- Asegurar que un proceso no pueda acceder a la memoria que tiene asignada otro proceso.
- Mantener un control sobre las zonas de memoria asignadas y las que no lo están.
- En los casos en los que la memoria principal esta totalmente ocupada, se encarga de realizar el intercambio (*swapping*) entre la memoria principal y la secundaria.

Se puede realizar la asignación de diferentes formas:

- Asignación en **particiones fijas**: el sistema operativo divide, lógicamente, la memoria disponible en particiones fijas en el tiempo (siempre del mismo tamaño, aunque no tienen por qué ser todas igual de grandes). Una vez hecho esto, cuando llegan los procesos, existen dos variantes:
  - Asignación con una sola cola por partición. En ella, que cada partición tiene una cola de procesos que se gestiona de manera independiente.
  - Asignación con una cola común a todas las particiones. En este caso, existe una única cola de procesos, que van siendo asignados a la mejor partición disponible en cada momento.
- Asignación en **particiones variables**: en esta variante, el sistema operativo genera las particiones y decide su tamaño tras la llegada del proceso, adaptando su tamaño a las necesidades del mismo. Aunque esta situación se aproxima más a la realidad —ya que el número de procesos y sus tamaños son variables, lo que permite aprovechar mejor la memoria total—, también da lugar a un modo de trabajo más complejo para el sistema operativo.
- **Memoria virtual**: cuando el proceso no cabe completamente (ni aún reservando toda la memoria para él), solo es posible ejecutarlo en el sistema si usamos la memoria virtual. Lo que hace el sistema operativo es que, a través de diferentes técnicas, divide la memoria que necesita el proceso en fragmentos que va cargando en la memoria principal según se

necesitan. Cuando la memoria está llena y se necesita más espacio, estos fragmentos se descargan de la memoria principal y se reservan en la memoria secundaria. Las técnicas empleadas para fragmentar el proceso son las siguientes:

- La paginación pura, que consiste en dividir la memoria principal en un conjunto de particiones (todas iguales) llamadas marcos de página. Cada proceso se divide en fragmentos del mismo tamaño llamados páginas. El sistema operativo va asignando marcos de página al proceso y va cargando en ellos las páginas del mismo, aunque no tienen por qué estar juntos.
- Por otro lado, la segmentación pura se basa en que el programador decide cómo dividir las partes de su proceso, que se cargarán en la memoria a modo de segmentos. A diferencia de las páginas, los segmentos sí pueden tener distintos tamaños.

C. Gestión de la entrada y la salida (E/S)

La gestión de la E/S es la responsable de facilitar el intercambio de información entre los periféricos y el usuario. Así, podemos encontrar periféricos de tres tipos:

- **Entrada**: llevan información desde el usuario hacia el sistema. Ratón, teclado, webcam.
- Salida: muestran información del sistema al usuario. Por ejemplo, la pantalla o la impresora.
- Entrada/**Salida**: realizan ambas funciones, como ocurre con los dispositivos de almacenamiento o de comunicaciones.

Dada esta gran variedad de periféricos, el sistema operativo utiliza, para simplificar al usuario la tarea de comunicación con dichos periféricos, los **controladores de dispositivos** (*device drivers*), que actúan como interfaz entre el usuario y el dispositivo.

Así, podemos encontrar elementos que funcionan como intermediarios y retienen la información hasta que el dispositivo —mucho más lento que el sistema operativo— es capaz de procesar dicha información. En este sentido, encontramos dos tipos:

- **Spools**: no admiten diferentes orígenes a la vez. Por ejemplo, una impresora no puede admitir parte de un trabajo hasta que finalice completamente el anterior.
- **Buffers**: sí admiten diferentes orígenes simultáneamente. Por ejemplo, las instrucciones de grabado o la lectura de un disco duro.

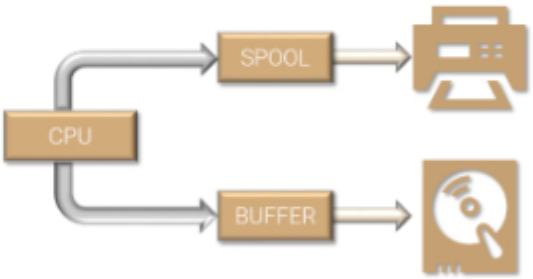

Figura 3.4. Spools y Buffers.

D. Gestión del Sistema de archivos

Conviene diferenciar la gestión que se lleva a cabo de los archivos y las carpetas a nivel físico —cuya forma de almacenar la información se encuentra estrechamente emparentada con el dispositivo concreto empleado a muy bajo nivel, lo que nos lleva a usar caras, pistas o sectores de un disco duro— con el nivel lógico, donde el sistema operativo abstraerá toda esta información para que el usuario la organice en archivos o carpetas.

Los **archivos** son el eje central de cualquier sistema de almacenamiento de información. Cada archivo consta de una serie de atributos que sirven para identificarlo y gestionarlo correctamente. Los atributos dependen del sistema de ficheros y, por tanto, del sistema operativo que usemos, pero generalmente podemos encontrar:

- Nombre: identificado del archivo.
- Extensión: caracteres que se colocan detrás del «.» en algunos sistemas operativos. Indican el tipo de archivo de que se trata. Ppor ejemplo, .txt designa archivos de texto, mientras que .mp3 se refiere a archivos de sonido.
- Permisos: indica quién puede leer, modificar, ejecutar… el archivo.
- Creador: quién creó el archivo originalmente.
- Fecha de creación: fecha y hora de creación del archivo.
- Fecha de última modificación: fecha y hora en la que se alteró el archivo por última vez.
- Tamaño actual: tamaño, en bytes, que ocupa el archivo.

Por otro lado, tenemos las **carpetas**, unos archivos especiales que sirven para organizar la información, puesto que pueden contener en su interior archivos u otras carpetas. De este modo, se consigue una estructura jerárquica en la que todos los archivos y todas las carpetas dependen de un nexo común llamado raíz, de la que depende todo el sistema de ficheros. Las carpetas también tienen sus propios atributos (coinciden, en gran parte, con los atributos de los archivos).

E. Carpetas especiales

Dentro del sistema de ficheros de la mayoría de sistemas operativos (incluidos los de Microsoft y GNU/Linux), existen dos carpetas especiales, que son:

- . (un punto): hace referencia a la carpeta en el que nos encontramos actualmente.
- .. (punto, punto): hace referencia a la carpeta padre de la carpeta actual, de la que depende.