**2.1. ¿Cuáles son los tres objetivos de diseño de un sistema operativo?**

Un sistema operativo es un programa que controla la ejecución de aplicaciones y programas y que actúa como interfaz entre las aplicaciones y el hardware del PC. Se puede considerar que un sistema operativo tiene los siguientes tres objetivos:

* **Facilidad de uso.** Un sistema operativo facilita el uso de un computador.
* **Eficiencia.** Un sistema operativo permite que los recursos de un sistema de computación se puedan utilizar de una manera eficiente.
* **Capacidad para evolucionar.** Un sistema operativo se debe construir de tal forma que se puedan desarrollar, probar e introducir nuevas funciones en el sistema sin interferir con su servicio.

**2.2. ¿Qué es el núcleo de un sistema operativo?**

Un programa de usuario ejecuta en **modo usuario**, en el cual los usuarios no pueden acceder a ciertas áreas de memoria y no puede ejecutar ciertas instrucciones. El sistema operativo ejecuta en modo sistema, o lo que se denomina **modo núcleo**, en el cual se pueden ejecutar instrucciones privilegiadas y se puede acceder a áreas de memoria protegidas.

Una porción del sistema operativo se encuentra en la memoria principal. Esto incluye el ***kernel*,** o **núcleo,** que contiene las funciones del sistema operativo más frecuentemente utilizadas y, en cierto momento, otras porciones del sistema operativo actualmente en uso. El resto de la memoria principal contiene programas y datos de usuario. La asignación de este recurso (memoria principal) es controlada de forma conjunta por el sistema operativo y el hardware de gestión de memoria del procesador. El sistema operativo decide cuándo un programa en ejecución puede utilizar un dispositivo de E/S y controla el acceso y uso de los ficheros. El procesador es también un recurso, y el sistema operativo debe determinar cuánto tiempo de procesador debe asignarse a la ejecución de un programa de usuario particular. En el caso de un sistema multiprocesador, esta decisión debe ser tomada por todos los procesadores.

**2.3. ¿Qué es multiprogramación?**

La multiprogramación es una técnica utilizada en sistemas operativos que permite ejecutar múltiples programas de manera concurrente en una computadora, de manera que parece que están ejecutándose al mismo tiempo. Aunque en realidad el procesador ejecuta instrucciones de un programa a la vez, la multiprogramación logra la ilusión de concurrencia al alternar rápidamente entre los diferentes programas en ejecución. En otras palabras, La multiprogramación es una técnica sistemas operativos donde varios programas se cargan en la memoria simultáneamente. El CPU cambia rápidamente entre los programas, permitiendo la imitación de una ejecución simultánea.

La multiprogramación aprovecha los períodos de inactividad del procesador, como las esperas de E/S (Entrada/Salida) o los tiempos de espera de memoria, para ejecutar otros programas en segundo plano. De esta manera, se optimiza el uso de los recursos del sistema y se aumenta la eficiencia del procesador, permitiendo que múltiples usuarios o tareas compartan la misma computadora de manera efectiva.

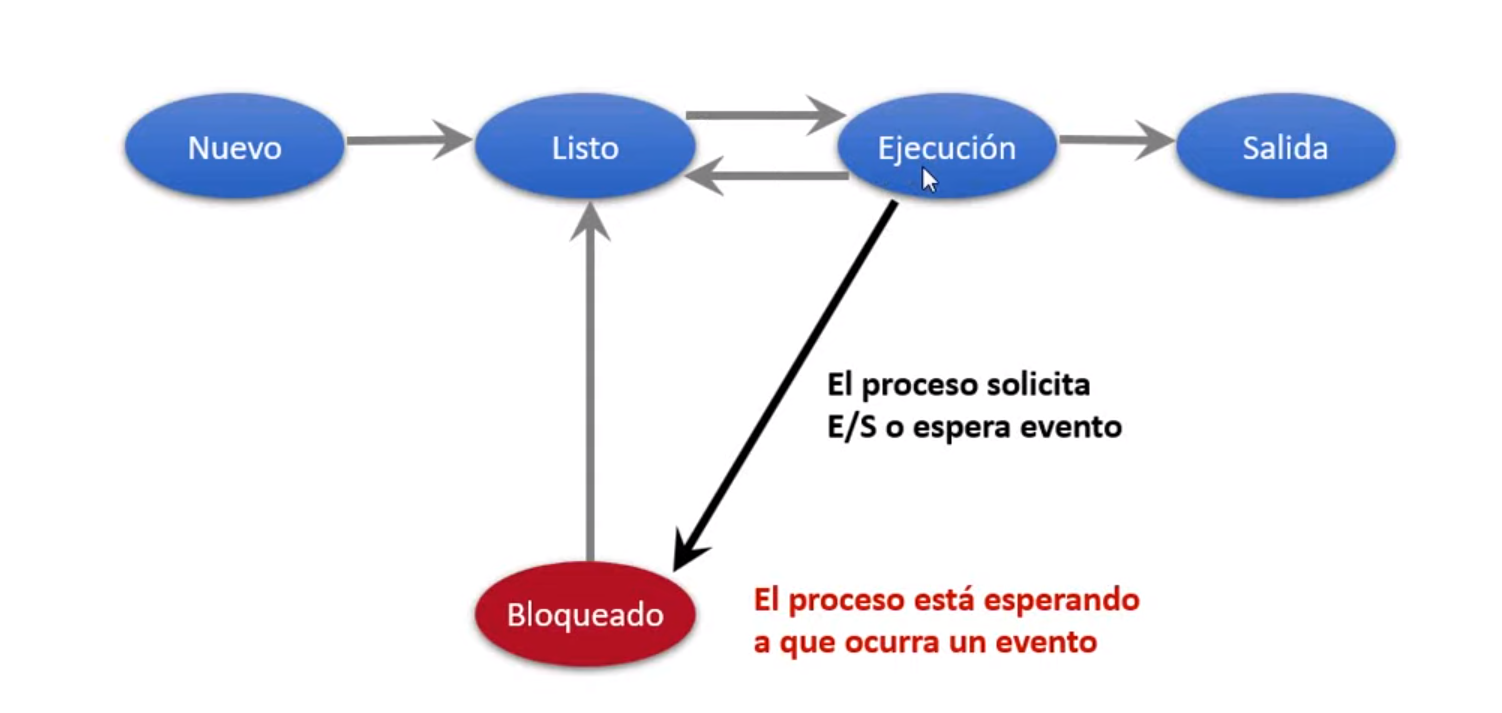
**2.4. ¿Qué es un proceso**

* Un programa en ejecución.
* Una instancia de un programa ejecutándose en un computador.
* La entidad que se puede asignar o ejecutar en un procesador.
* Una unidad de actividad caracterizada por un solo hilo secuencial de ejecución, un estado actual, y un conjunto de recursos del sistema asociados.

Un proceso es un programa en ejecución en un sistema operativo. Puede ser una aplicación de usuario, un servicio del sistema, o incluso parte del propio sistema operativo. Cada proceso tiene su propio espacio de memoria asignado, que incluye el código ejecutable, los datos, la pila de ejecución y otras secciones necesarias para su funcionamiento.

Los procesos pueden interactuar entre sí y con el sistema operativo mediante el intercambio de datos, señales y solicitudes de servicios. Cada proceso se ejecuta de forma independiente de otros procesos, y el sistema operativo se encarga de gestionar los recursos compartidos, como la CPU, la memoria y los dispositivos de E/S, para garantizar una ejecución segura y eficiente.

Además, los procesos pueden tener diferentes estados, como en ejecución, listo, bloqueado o terminado, dependiendo de su actividad y necesidades de recursos en un momento dado. La gestión de procesos es una parte fundamental de los sistemas operativos modernos, ya que permite la multitarea y la ejecución concurrente de múltiples programas en un mismo sistema.



Lo que se necesita para enfrentarse a estos problemas es una forma sistemática de monitorizar y controlar la ejecución de varios programas en el procesador. El concepto de proceso proporciona los fundamentos. Se puede considerar que un proceso está formado por los siguientes tres componentes:

* Un programa ejecutable.
* Los datos asociados que necesita el programa (variables, espacio de trabajo, *buffers*, etc.).
* El contexto de ejecución del programa.

**2.5. ¿Cómo utiliza el sistema operativo el contexto de ejecución de un proceso?**

El **contexto de ejecución**, o **estado del proceso**, es el conjunto de datos interno por el cual el sistema operativo es capaz de supervisar y controlar el proceso. Esta información interna está separada del proceso, porque el sistema operativo tiene información a la que el proceso no puede acceder. El contexto incluye toda la información que el sistema operativo necesita para gestionar el proceso y que el procesador necesita para ejecutar el proceso apropiadamente. El contexto incluye el contenido de diversos registros del procesador, tales como el contador de programa y los registros de datos. También incluye información de uso del sistema operativo, como la prioridad del proceso y si un proceso está esperando por la finalización de un evento de E/S particular.

El proceso puede verse como una estructura de datos. Un proceso puede estar en ejecución o esperando ejecutarse. El **estado** completo del proceso en un instante dado se contiene en su contexto. Esta estructura permite el desarrollo de técnicas potentes que aseguran la coordinación y la cooperación entre los procesos. Se pueden diseñar e incorporar nuevas características en el sistema operativo (por ejemplo, la prioridad), expandiendo el contexto para incluir cualquier información nueva que se utilice para dar soporte a dicha característica.

**2.6. Liste y explique brevemente cinco responsabilidades relacionadas con la gestión de almacenamiento de un sistema operativo típico.**

**GESTIÓN DE MEMORIA**

Un entorno de computación que permita programación modular y el uso flexible de los datos puede ayudar a resolver mejor las necesidades de los usuarios. Los gestores de sistema necesitan un control eficiente y ordenado de la asignación de los recursos. Para satisfacer estos requisitos, el sistema operativo tiene cinco responsabilidades principales de gestión de almacenamiento:

1. **Aislamiento de procesos.** El sistema operativo debe evitar que los procesos independientes interfieran en la memoria de otro proceso, tanto datos como instrucciones.
2. **Asignación y gestión automática.** Los programas deben tener una asignación dinámica de memoria por demanda, en cualquier nivel de la jerarquía de memoria. La asignación debe ser transparente al programador. Por tanto, el programador no debe preocuparse de aspectos relacionados con limitaciones de memoria, y el sistema operativo puede lograr incrementar la eficiencia, asignando memoria a los trabajos sólo cuando se necesiten.
3. **Soporte a la programación modular.** Los programadores deben ser capaces de definir módulos de programación y crear, destruir, y alterar el tamaño de los módulos dinámicamente.
4. **Protección y control de acceso.** La compartición de memoria, en cualquier nivel de la jerarquía de memoria, permite que un programa direccione un espacio de memoria de otro proceso. Esto es deseable cuando se necesita la compartición por parte de determinadas aplicaciones. Otras veces, esta característica amenaza la integridad de los programas e incluso del propio sistema operativo. El sistema operativo debe permitir que varios usuarios puedan acceder de distintas formas a porciones de memoria.
5. **Almacenamiento a largo plazo.** Muchas aplicaciones requieren formas de almacenar la información durante largos periodos de tiempo, después de que el computador se haya apagado.

Normalmente, los sistemas operativos alcanzan estos requisitos a través del uso de la memoria virtual y las utilidades de los sistemas operativos. El sistema operativo implementa un almacenamiento a largo plazo, con la información almacenada en objetos denominados ficheros. El fichero es un concepto lógico, conveniente para el programador y es una unidad útil de control de acceso y protección para los sistemas operativos.

**2.7. Explique la distinción entre una dirección real y una dirección virtual.**

La memoria virtual es una utilidad que permite a los programas direccionar la memoria desde un punto de vista lógico, sin importar la cantidad de memoria principal física disponible. La memoria virtual fue concebida como un método para tener múltiples trabajos de usuario residiendo en memoria principal de forma concurrente, de forma que no exista un intervalo de tiempo de espera entre la ejecución de procesos sucesivos, es decir, mientras un proceso se escribe en almacenamiento secundario y se lee el proceso sucesor. Debido a que los procesos varían de tamaño, si el procesador planifica un determinado número de procesos, es difícil almacenarlos compactamente en memoria principal. Se introdujeron los sistemas de paginación, que permiten que los procesos se compriman en un número determinado de bloques de tamaño fijo, denominados páginas. Un programa referencia una palabra por medio de una **dirección virtual**, que consiste en un número de página y un desplazamiento dentro de la página. Cada página de un proceso se puede localizar en cualquier sitio de memoria principal. El sistema de paginación proporciona una proyección dinámica entre las direcciones virtuales utilizadas en el programa y una **dirección real**, o dirección física, de memoria principal. Mejor explicado en el siguiente link:

<https://forum.huawei.com/enterprise/es/%C2%BFQu%C3%A9-es-la-memoria-virtual-en-el-sistema-operativo/thread/667239728874930176-667212885295771648>

*Conclusión:*

* **Dirección Real**: Es la ubicación física exacta en la memoria física de la computadora. Es como una dirección de casa en un mapa: te lleva directamente al lugar específico donde se encuentra la información en la memoria física de la computadora.
* **Dirección Virtual**: Es una dirección lógica o imaginaria utilizada por los programas. En lugar de referirse directamente a una ubicación física en la memoria, se refiere a una ubicación abstracta que luego se traduce en una dirección real por el sistema operativo. Es como una dirección postal que te guía a una ubicación general, pero el correo se entrega a una dirección real específica. La memoria virtual permite que los programas se ejecuten como si tuvieran acceso a toda la memoria física, aunque en realidad se utiliza solo una parte de ella. Esto es útil cuando se tienen múltiples programas en ejecución y se necesita administrar eficientemente la memoria disponible. El hardware llamado MMU es el encargado de traducir las direcciones virtuales en direcciones reales.

La dirección virtual sería como la dirección de código postal, que es la dirección lógica o abstracta utilizada por el programa. Mientras que la dirección real sería como la dirección física de una casa, que es la ubicación exacta en la memoria física de la computadora. Cuando el programa necesita acceder a una parte específica de su código o datos en la memoria, el sistema operativo se encarga de traducir esa dirección virtual en una dirección real, permitiendo así el acceso a la información almacenada en la memoria principal. Esto es fundamental para garantizar que los programas puedan ejecutarse de manera eficiente y sin conflictos en un entorno de memoria compartida.

**2.8. Describa la técnica de planificación round-robin o turno rotatorio.**

Una responsabilidad clave de los sistemas operativos es la gestión de varios recursos disponibles para ellos (espacio de memoria principal, dispositivos de E/S, procesadores) y para planificar su uso por parte de los distintos procesos activos.

La planificación y la gestión de recursos son esencialmente problemas de investigación, y se pueden aplicar los resultados matemáticos de esta disciplina. Adicionalmente, medir la actividad del sistema es importante para ser capaz de monitorizar el rendimiento y realizar los ajustes correspondientes.

La Figura 2.11 sugiere los principales elementos del sistema operativo relacionados con la planificación de procesos y la asignación de recursos en un entorno de multiprogramación. El sistema operativo mantiene un número de colas, cada una de las cuales es simplemente una lista de procesos esperando por algunos recursos. La cola a corto plazo está compuesta por procesos que se encuentran en memoria principal (o al menos una porción mínima esencial de cada uno de ellos está en memoria principal) y están listos para ejecutar, siempre que el procesador esté disponible. Cualquiera de estos procesos podría usar el procesador a continuación. Es responsabilidad del planificador a corto plazo, o *dispatcher*, elegir uno de ellos. Una estrategia común es asignar en orden a cada proceso de la cola un intervalo de tiempo; esta técnica se conoce como ***round-robin*** o **turno rotatorio**. En efecto, la técnica de turno rotatorio emplea una cola circular. Otra estrategia consiste en asignar niveles de prioridad a los distintos procesos, siendo el planificador el encargado de elegir los procesos en orden de prioridad.

La cola a largo plazo es una lista de nuevos trabajos esperando a utilizar el procesador. El sistema operativo añade trabajos al sistema transfiriendo un proceso desde la cola a largo plazo hasta la cola a corto plazo. En este punto, se debe asignar una porción de memoria principal al proceso entrante. Por tanto, el sistema operativo debe estar seguro de que no sobrecarga la memoria o el tiempo de procesador admitiendo demasiados procesos en el sistema. Hay una cola de E/S por cada dispositivo de E/S. Más de un proceso puede solicitar el uso del mismo dispositivo de E/S. Todos los procesos que esperan utilizar dicho dispositivo, se encuentran alineados en la cola del dispositivo. De nuevo, el sistema operativo debe determinar a qué proceso le asigna un dispositivo de E/S disponible.

*Conclusión:*

La técnica de planificación round-robin, o turno rotatorio, es una estrategia utilizada por los sistemas operativos para asignar tiempo de procesador a los procesos en ejecución de manera equitativa y justa.

Imagina que tienes una lista de tareas por hacer y quieres asegurarte de que todas tengan la misma oportunidad de ser completadas. En lugar de enfocarte en una tarea a la vez, decides dividir tu tiempo entre todas las tareas de manera igualitaria. Esto es lo que hace el planificador round-robin.

En un sistema operativo, los procesos listos para ejecutar se colocan en una cola. El planificador round-robin toma esta cola y asigna un pequeño intervalo de tiempo de procesador a cada proceso en un orden circular. Cada proceso obtiene una rebanada de tiempo de CPU para realizar su trabajo antes de pasar al siguiente proceso en la cola. Una vez que todos los procesos han recibido su turno, el planificador vuelve al principio de la cola y comienza de nuevo, rotando continuamente a través de los procesos disponibles.

Esta técnica asegura que ningún proceso monopolice el tiempo de CPU y que todos los procesos tengan la oportunidad de avanzar en su ejecución de manera justa. Es como compartir un pastel equitativamente entre amigos: todos reciben una porción antes de que se repita el ciclo.

*Además:*

La cola a corto y largo plazo son estructuras utilizadas por los sistemas operativos para gestionar los procesos en ejecución y su acceso a los recursos del sistema.

- **Cola a corto plazo**: Esta cola contiene los procesos que están listos para ser ejecutados en la CPU. Estos procesos ya están en la memoria principal y solo están esperando su turno para ser ejecutados por el procesador. El planificador a corto plazo, también conocido como dispatcher, elige de manera eficiente qué proceso se ejecutará a continuación. En el contexto de la técnica de planificación round-robin, esta cola sería utilizada para almacenar los procesos que están listos para recibir una rebanada de tiempo de CPU.

- **Cola a largo plazo**: Por otro lado, la cola a largo plazo contiene los procesos que aún no están en memoria principal y están esperando ser admitidos en el sistema para su ejecución. Estos pueden ser procesos recién creados o procesos que previamente se habían suspendido o terminado y que ahora deben ser cargados en memoria. El sistema operativo debe decidir cuántos de estos procesos admitir en la memoria principal en función de la disponibilidad de recursos y evitar la sobrecarga del sistema.

**2.9. Explique la diferencia entre un núcleo monolítico y un micronúcleo.**

Hasta hace relativamente poco tiempo, la mayoría de los sistemas operativos estaban formados por un gran **núcleo monolítico**. Estos grandes núcleos proporcionan la mayoría de las funcionalidades consideradas propias del sistema operativo, incluyendo la planificación, los sistemas de ficheros, las redes, los controladores de dispositivos, la gestión de memoria y otras funciones. Normalmente, un núcleo monolítico se implementa como un único proceso, con todos los elementos compartiendo el mismo espacio de direcciones. Una **arquitectura micronúcleo** asigna sólo unas pocas funciones esenciales al núcleo, incluyendo los espacios de almacenamiento, comunicación entre procesos (IPC), y la planificación básica. Ciertos procesos proporcionan otros servicios del sistema operativo, algunas veces denominados servidores, que ejecutan en modo usuario y son tratados como cualquier otra aplicación por el micronúcleo. Esta técnica desacopla el núcleo y el desarrollo del servidor. Los servidores pueden configurarse para aplicaciones específicas o para determinados requisitos del entorno. La técnica micronúcleo simplifica la implementación, proporciona flexibilidad y se adapta perfectamente a un entorno distribuido. En esencia, un micronúcleo interactúa con procesos locales y remotos del servidor de la misma forma, facilitando la construcción de los sistemas distribuidos.

*Conclusión:*

Imagina que el sistema operativo es como una gran ciudad y el núcleo es el centro de operaciones, donde se toman todas las decisiones importantes. Ahora, hay dos formas diferentes de organizar este centro:

- **Núcleo Monolítico:** Es como tener un mega-edificio que alberga todas las funciones importantes del sistema operativo: desde la planificación y la gestión de memoria hasta las redes y los controladores de dispositivos. Todo está integrado en un solo lugar, compartiendo el mismo espacio y recursos. Es como tener un edificio enorme donde viven todas las funciones del sistema operativo juntas.

- **Micronúcleo:** Ahora, imagina que en lugar de un mega-edificio, tienes un pequeño núcleo que solo se encarga de las funciones esenciales: almacenamiento, comunicación entre procesos y planificación básica. Todo lo demás, como las redes, los sistemas de archivos y los controladores de dispositivos, se manejan por separado, como edificios independientes llamados "servidores". Estos servidores funcionan como aplicaciones normales, pero están especializados en ciertas tareas del sistema operativo. Es como tener un pequeño núcleo central que coordina diferentes servicios, cada uno en su propio edificio especializado.

Entonces, la diferencia clave es que el núcleo monolítico tiene todo integrado en un solo lugar, mientras que el micronúcleo divide las funciones en partes más pequeñas y flexibles, lo que permite una mayor personalización y adaptabilidad, especialmente en entornos distribuidos.

**2.10. ¿En qué consiste el uso de multihilos o multithreading?**

El ***Multitheading*** es una técnica en la cual un proceso, ejecutando una aplicación, se divide en una serie de hilos o *threads* que pueden ejecutar concurrentemente. Se pueden hacer las siguientes distinciones:

* ***Thread* o hilo.** Se trata de una unidad de trabajo. Incluye el contexto del procesador (que contiene el contador del programa y el puntero de pila) y su propia área de datos para una pila (para posibilitar el salto a subrutinas). Un hilo se ejecuta secuencialmente y se puede interrumpir de forma que el procesador pueda dar paso a otro hilo.
* **Proceso.** Es una colección de uno o más hilos y sus recursos de sistema asociados (como la memoria, conteniendo tanto código, como datos, ficheros abiertos y dispositivos). Esto corresponde al concepto de programa en ejecución. Dividiendo una sola aplicación en múltiples hilos, el programador tiene gran control sobre la modularidad de las aplicaciones y la temporización de los eventos relacionados con la aplicación.

La técnica *multithreading* es útil para las aplicaciones que llevan a cabo un número de tareas esencialmente independientes que no necesitan ser serializadas. Un ejemplo es un servidor de bases de datos que escucha y procesa numerosas peticiones de cliente. Con múltiples hilos ejecutándose dentro del mismo proceso, intercambiar la ejecución entre los hilos supone menos sobrecarga del procesador que intercambiar la ejecución entre diferentes procesos pesados. Los hilos son también útiles para estructurar procesos que son parte del núcleo del sistema operativo.

*Conclusión:*

El uso de multihilos o multithreading consiste en la capacidad de un programa para realizar múltiples tareas simultáneamente dentro de un solo proceso. Los hilos son como subprocesos dentro de un programa que pueden ejecutar diferentes partes del código de forma independiente. Esto significa que un programa puede realizar múltiples acciones al mismo tiempo, como procesar datos, manejar la interfaz de usuario y realizar operaciones de red, todo dentro del mismo proceso. Esto es especialmente útil en aplicaciones que requieren una respuesta rápida y que tienen múltiples tareas que se pueden realizar en paralelo.