**1.1. Enumere y defina brevemente los cuatro elementos principales de un computador.**

Al más alto nivel, un computador consta del procesador, la memoria y los componentes de E/S, incluyendo uno o más módulos de cada tipo. Estos componentes se interconectan de manera que se pueda lograr la función principal del computador, que es ejecutar programas. Por tanto, hay cuatro elementos estructurales principales:

* **Procesador.** Controla el funcionamiento del computador y realiza sus funciones de procesamiento de datos. Cuando sólo hay un procesador, se denomina usualmente **unidad central de proceso** (*Central Processing Unit*, CPU).
* **Memoria principal.** Almacena datos y programas. Esta memoria es habitualmente volátil; es decir, cuando se apaga el computador, se pierde su contenido. En contraste, el contenido de la memoria del disco se mantiene incluso cuando se apaga el computador. A la memoria principal se le denomina también *memoria real* o *memoria primaria*.
* **Módulos de E/S.** Transfieren los datos entre el computador y su entorno externo. El entorno externo está formado por diversos dispositivos, incluyendo dispositivos de memoria secundaria (por ejemplo, discos), equipos de comunicaciones y terminales.
* **Bus del sistema.** Proporciona comunicación entre los procesadores, la memoria principal y los módulos de E/S.

**1.2. Defina las dos categorías principales de los registros del procesador.**

Un procesador incluye un conjunto de registros que proporcionan un tipo de memoria que es más rápida y de menor capacidad que la memoria principal. Los registros del procesador sirven para dos funciones:

* **Registros visibles para el usuario.** Permiten al programador en lenguaje máquina o en ensamblador minimizar las referencias a memoria principal optimizando el uso de registros. Para lenguajes de alto nivel, un compilador que realice optimización intentará tomar decisiones inteligentes sobre qué variables se asignan a registros y cuáles a posiciones de memoria principal. Algunos lenguajes de alto nivel, tales como C, permiten al programador sugerir al compilador qué variables deberían almacenarse en registros.
* **Registros de control y estado.** Usados por el procesador para controlar su operación y por rutinas privilegiadas del sistema operativo para controlar la ejecución de programas.

No hay una clasificación nítida de los registros entre estas dos categorías. Por ejemplo, en algunas máquinas el contador de programa es visible para el usuario, pero en muchas otras no lo es. Sin embargo, para el estudio que se presenta a continuación, es conveniente utilizar estas categorías.

**1.3. En términos generales, ¿cuáles son las cuatro acciones distintas que puede especificar una instrucción de máquina?**

Un programa que va a ejecutarse en un procesador consta de un conjunto de instrucciones almacenado en memoria. En su forma más simple, el procesamiento de una instrucción consta de dos pasos: el procesador lee (*busca*) instrucciones de la memoria, una cada vez, y ejecuta cada una de ellas. La ejecución del programa consiste en repetir el proceso de búsqueda y ejecución de instrucciones. La ejecución de la instrucción puede involucrar varias operaciones dependiendo de la naturaleza de la misma.

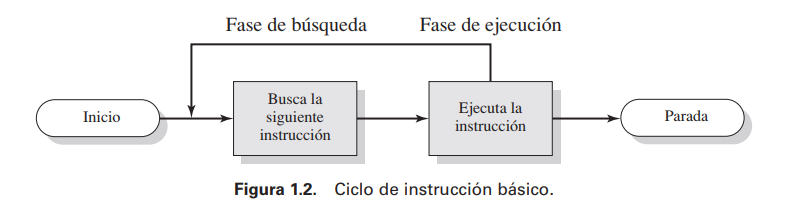
Se denomina *ciclo de instrucción* al procesamiento requerido por una única instrucción.

Al principio de cada ciclo de instrucción, el procesador lee una instrucción de la memoria. En un procesador típico, el contador del programa (PC) almacena la dirección de la siguiente instrucción que se va a leer. A menos que se le indique otra cosa, el procesador siempre incrementa el PC después de cada instrucción ejecutada, de manera que se leerá la siguiente instrucción en orden secuencial (es decir, la instrucción situada en la siguiente dirección de memoria más alta).

La instrucción leída se carga dentro de un registro del procesador conocido como registro de instrucción (IR). La instrucción contiene bits que especifican la acción que debe realizar el procesador. El procesador interpreta la instrucción y lleva a cabo la acción requerida. En general, estas acciones se dividen en cuatro categorías:

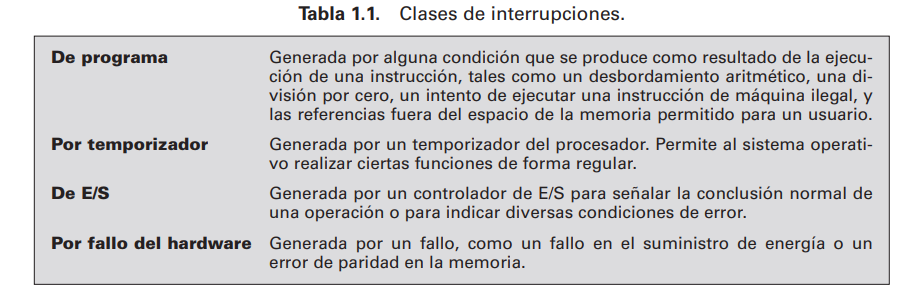
* **Procesador-memoria.** Se pueden transferir datos desde el procesador a la memoria o viceversa.
* **Procesador-E/S.** Se pueden enviar datos a un dispositivo periférico o recibirlos desde el mismo, transfiriéndolos entre el procesador y un módulo de E/S.
* **Procesamiento de datos.** El procesador puede realizar algunas operaciones aritméticas o lógicas sobre los datos.
* **Control.** Una instrucción puede especificar que se va a alterar la secuencia de ejecución. Por ejemplo, el procesador puede leer una instrucción de la posición 149, que especifica que la siguiente instrucción estará en la posición 182. El procesador almacenará en el contador del programa un valor de 182. Como consecuencia, en la siguiente fase de búsqueda, se leerá la instrucción de la posición 182 en vez de la 150.

Una ejecución de una instrucción puede involucrar una combinación de estas acciones.



**1.4. ¿Qué es una interrupción?**

Una interrupción es un mecanismo esencial en los computadores que permite a otros módulos, como la memoria y los dispositivos de entrada/salida (E/S), interrumpir la secuencia normal de ejecución del procesador. Este sistema se implementa para mejorar la eficiencia del procesador y optimizar su tiempo de uso. Por ejemplo, dado que la velocidad de procesamiento del procesador supera con creces la velocidad de algunos dispositivos de E/S, como las impresoras, el procesador puede quedarse inactivo durante períodos prolongados esperando a que se completen las operaciones de E/S. Las interrupciones permiten que el procesador atienda otras tareas mientras espera la finalización de estas operaciones más lentas, lo que maximiza la utilización de sus recursos y minimiza los tiempos de inactividad.



**1.5. ¿Cómo se tratan múltiples interrupciones?**

El estudio realizado hasta el momento ha tratado solamente el caso de que se produzca una única interrupción. Supóngase, sin embargo, que se producen múltiples interrupciones. Por ejemplo, un programa puede estar recibiendo datos de una línea de comunicación e imprimiendo resultados al mismo tiempo. La impresora generará una interrupción cada vez que completa una operación de impresión. El controlador de la línea de comunicación generará una interrupción cada vez que llega una unidad de datos. La unidad podría consistir en un único carácter o en un bloque, dependiendo de la naturaleza del protocolo de comunicaciones. En cualquier caso, es posible que se produzca una interrupción de comunicación mientras se está procesando una interrupción de la impresora.

Cuando se enfrenta con múltiples interrupciones, existen dos alternativas principales para su tratamiento:

La primera opción consiste en *inhabilitar las interrupciones* durante el procesamiento de una interrupción. Esto implica que el procesador ignora cualquier nueva señal de petición de interrupción mientras está ocupado manejando una interrupción previa. Sin embargo, esta estrategia no considera la prioridad o urgencia relativa de las interrupciones, lo que podría llevar a la pérdida de datos importantes si las operaciones de E/S no se procesan rápidamente.

La segunda estrategia implica *asignar prioridades* a las interrupciones y permitir que una interrupción de mayor prioridad interrumpa la ejecución de una de menor prioridad. Por ejemplo, en un sistema con múltiples dispositivos de E/S, cada uno tiene una prioridad asignada y se atienden en orden de prioridad. Esta estrategia garantiza que las interrupciones más urgentes se manejen primero, evitando la pérdida de datos y asegurando una respuesta rápida a eventos críticos.

**1.6. ¿Qué características distinguen a los diversos elementos de una jerarquía de memoria?**

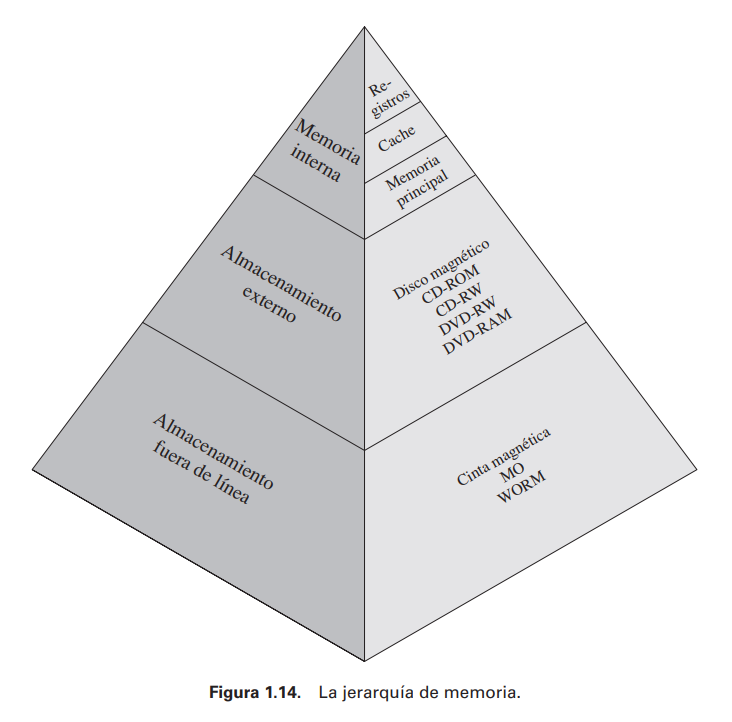
La pregunta sobre cuánta debe ser su capacidad es algo que no tiene límite. Si se dispone de una determinada capacidad, probablemente se desarrollarán aplicaciones que la usarán. La cuestión acerca de la velocidad tiene, hasta cierto tiempo, una respuesta más fácil. Para alcanzar un rendimiento máximo, la memoria debe ser capaz de mantener el ritmo del procesador. Es decir, según el procesador va ejecutando instrucciones, no debería haber pausas esperando que estén disponibles las instrucciones o los operandos. Se debe considerar también la última pregunta. Para un sistema práctico, el coste de la memoria debe ser razonable en relación con los otros componentes.

Como se podría esperar, hay un compromiso entre las tres características fundamentales de la memoria: a saber, coste, capacidad y tiempo de acceso. En cualquier momento dado, se utilizan diversas tecnologías para implementar los sistemas de memoria. En todo este espectro de tecnologías, se cumplen las siguientes relaciones:

* Cuanto menor tiempo de acceso, mayor coste por bit.
* Cuanto mayor capacidad, menor coste por bit.
* Cuanto mayor capacidad, menor velocidad de acceso.

La solución a este dilema consiste en no basarse en un único componente de memoria o en una sola tecnología, sino emplear una **jerarquía de memoria**. En la Figura 1.14 se muestra una jerarquía típica. Según se desciende en la jerarquía, ocurre lo siguiente:

1. Disminución del coste por bit.
2. Aumento de la capacidad.
3. Aumento del tiempo de acceso.
4. Disminución de la frecuencia de acceso a la memoria por parte del procesador.



**1.7. ¿Qué es una memoria cache?**

Una memoria caché es una memoria de acceso rápido que actúa como un intermediario entre el procesador y la memoria principal de un sistema computacional. Su objetivo es reducir el tiempo de acceso a los datos al proporcionar una ubicación de almacenamiento más cercana al procesador y de mayor velocidad que la memoria principal.

La necesidad de una memoria caché surge debido a la discrepancia en la velocidad entre el procesador y la memoria principal. A medida que la velocidad del procesador ha aumentado constantemente a lo largo de los años, la velocidad de acceso a la memoria principal ha permanecido relativamente más lenta. Para superar esta limitación, se emplea el principio de la proximidad, donde una memoria caché más pequeña y rápida contiene copias de partes críticas de la memoria principal.

Cuando el procesador necesita acceder a datos, primero verifica si están almacenados en la caché. Si se encuentran en la caché (lo que se conoce como un "acceso caché hit"), los datos se entregan al procesador rápidamente. En caso contrario, se realiza una lectura de la memoria principal y se carga un bloque de datos en la caché. Esta estrategia aprovecha la proximidad de referencias, ya que es probable que se acceda a datos cercanos en el tiempo nuevamente en el futuro.

**1.8. Enumere y defina brevemente las tres técnicas para las operaciones de E/S.**

Las tres técnicas para llevar a cabo operaciones de E/S son:

1. **E/S programada:** En esta técnica, el procesador es responsable de extraer los datos de la memoria principal en una operación de salida y de almacenarlos en ella en una operación de entrada. El software de E/S se escribe de manera que el procesador ejecuta instrucciones que le dan control directo sobre la operación de E/S, incluyendo la verificación del estado del dispositivo, el envío de mandatos de lectura o escritura, y la transferencia de los datos.

2. **E/S dirigida por interrupciones:** Esta técnica aborda el problema de la E/S programada, donde el procesador debe esperar inactivamente hasta que el módulo de E/S esté listo para recibir o enviar más datos. En la E/S dirigida por interrupciones, el procesador genera un mandato de E/S y continúa realizando otras tareas. El módulo de E/S interrumpe más tarde al procesador cuando está listo para intercambiar datos. Esta técnica es más eficiente que la E/S programada, ya que elimina la espera innecesaria del procesador.

3. **Acceso directo a memoria (DMA):** En el acceso directo a memoria, un módulo de DMA se encarga de la transferencia de datos entre la memoria y un dispositivo de E/S sin la intervención activa del procesador. El procesador delega la operación de E/S al módulo de DMA, que se encarga de la transferencia del bloque completo de datos, palabra por palabra, hacia o desde la memoria sin pasar por el procesador. Aunque el procesador está involucrado al principio y al final de la transferencia, durante la transferencia en sí, el procesador puede continuar con otras tareas. Aunque puede haber competencia por el uso del bus entre el módulo de DMA y el procesador, el DMA es mucho más eficiente para transferencias de E/S de múltiples palabras en comparación con las técnicas dirigidas por interrupciones o programadas.

**1.9. ¿Cuál es la diferencia entre la proximidad espacial y la temporal?**

La **proximidad espacial** se refiere a la tendencia de una ejecución a involucrar posiciones de memoria que están agrupadas. Esto refleja la tendencia de un procesador a acceder secuencialmente a las instrucciones. La proximidad espacial también refleja la tendencia de un programa a acceder de forma secuencial a las posiciones de datos, como cuando se procesa una tabla de datos.

La **proximidad temporal** hace referencia a la tendencia de un procesador a acceder a posiciones de memoria que se han utilizado recientemente. Por ejemplo, cuando se ejecuta un bucle, el procesador ejecuta el mismo juego de instrucciones repetidamente.

Tradicionalmente, la proximidad temporal se explota manteniendo en la memoria cache los valores de las instrucciones y los datos usados recientemente aprovechando una jerarquía de cache. La proximidad espacial se explota generalmente utilizando bloques de cache más grandes e incorporando mecanismos de lectura anticipada (se buscan elementos cuyo uso se prevé) en la lógica de control de la cache. **🡪 1.10**

**1.10. En general, ¿cuáles son las estrategias para aprovechar la proximidad espacial y la temporal?**

Para aprovechar la proximidad espacial y temporal en el diseño de sistemas de memoria y caché, se emplean diversas estrategias:

1. **Proximidad Espacial:**

- **Bloques de caché más grandes:** Al aumentar el tamaño de los bloques de caché, se incrementa la probabilidad de que se almacenen datos adyacentes en la memoria principal en la caché. Esto mejora la eficacia de la caché al explotar la proximidad espacial, ya que es más probable que se acceda a datos cercanos en la memoria principal simultáneamente.

**- Lectura anticipada (prefetching):** Se anticipa la necesidad de datos futuros y se los carga en la caché antes de que se soliciten explícitamente por el procesador. Esto se basa en patrones de acceso históricos o algoritmos predictivos para identificar datos que probablemente se necesitarán en el futuro cercano.

2. **Proximidad Temporal:**

- **Mantenimiento de valores recientes en caché**: Los datos e instrucciones utilizados recientemente se mantienen en la caché para aprovechar la proximidad temporal. Esto permite un acceso más rápido a estos datos si son requeridos nuevamente en un futuro cercano.

- **Políticas de reemplazo inteligente:** Se utilizan algoritmos de reemplazo en la caché que tienen en cuenta la frecuencia y recencia de acceso a los datos. Por ejemplo, el algoritmo LRU (Least Recently Used) reemplaza los datos menos recientes utilizados en la caché, lo que ayuda a mantener los datos más recientes en la caché para aprovechar la proximidad temporal.

En resumen, tanto la proximidad espacial como la temporal se aprovechan mediante el diseño de cachés con políticas de almacenamiento y reemplazo que optimizan el acceso a los datos. El objetivo es maximizar el rendimiento del sistema al minimizar los tiempos de acceso a la memoria y aumentar la eficiencia de la caché.

**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.

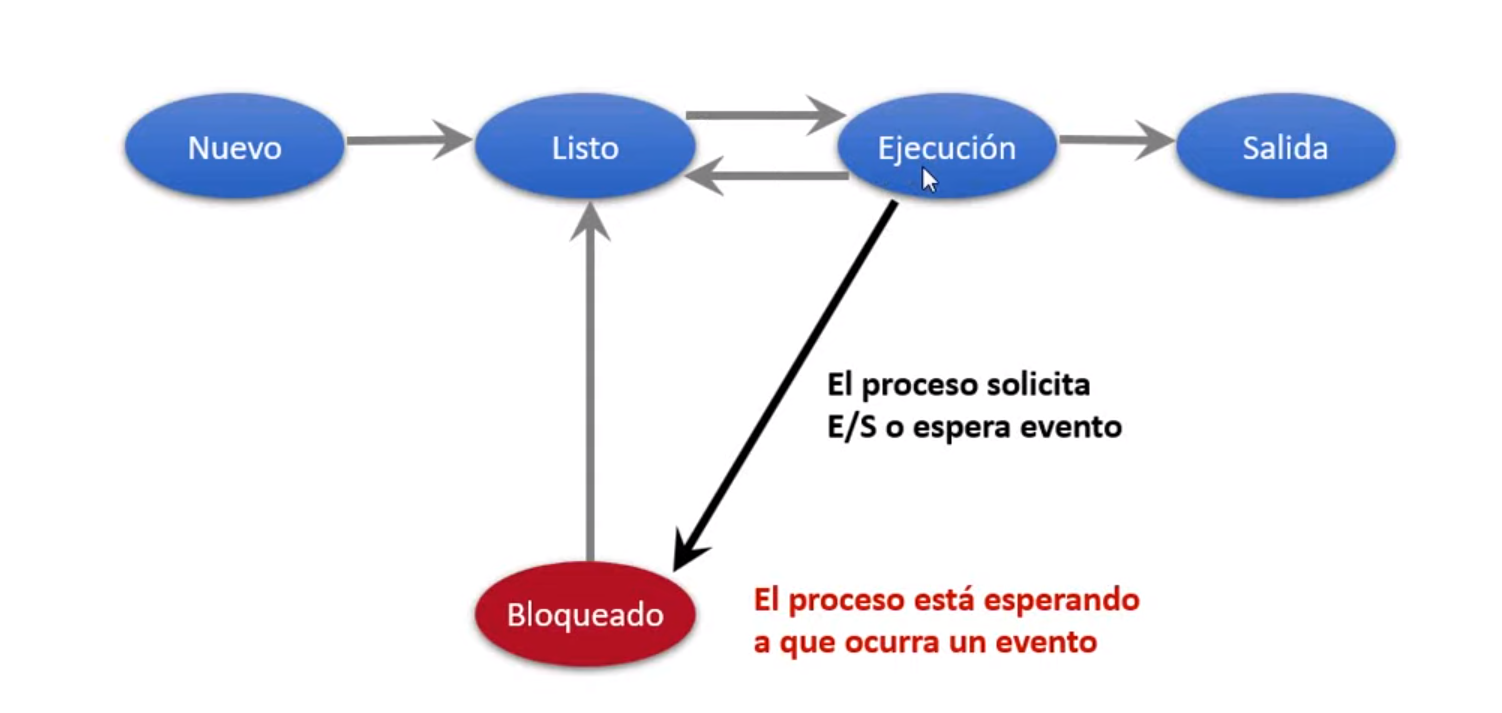
**2.4. ¿Qué es un proceso?** 🡪 Mas información en Resumen Unidad 1

* 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.

*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.

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.

En resumen, la cola a corto plazo maneja los procesos que ya están en memoria y listos para ser ejecutados, mientras que la cola a largo plazo maneja los procesos que están esperando ser cargados en memoria para su ejecución futura. Ambas colas son importantes para garantizar una gestión eficiente de los recursos del sistema y un equilibrio adecuado entre la concurrencia y la carga 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.