# RESUMEN TEORICO DE ISO

**Objetivos del resumen:**

**. Mejor entendimiento de la teoría y sobretodo poder responder mínimo 14 preguntas bien**

**Sistema de calificación de conocimiento:**

**. Luego de cada bloque de estudio evaluare mi conocimiento en un rango de preguntas esperado bien respuestas**

**Estrategia:**

**. Leer el resumen de la teoría, y los puntos que no sean claros repasarlos de la teoría original**

**. Una vez asentados los contenidos realizare las preguntas de repaso enviadas por la cátedra**

1) Si necesita tiempo ya que al ejecutarse varios procesos en simultaneo se necesita el multiplexeo del tiempo del CPU

Sí, el sistema operativo (SO) necesita tiempo de CPU para funcionar correctamente. El SO es el software que administra todos los recursos del hardware y proporciona servicios a las aplicaciones. Para realizar estas tareas, el SO debe ejecutar procesos y realizar operaciones, lo que requiere tiempo de CPU.

El SO se encarga de programar y asignar recursos a los procesos de las aplicaciones, administrar la memoria, manejar la entrada/salida, proporcionar seguridad, administrar archivos y sistemas de archivos, y mucho más. Todas estas tareas requieren tiempo de CPU para ejecutarse. Además, el SO también debe ocuparse de la planificación de procesos, lo que implica tomar decisiones sobre qué proceso se ejecutará en la CPU en un momento dado.

En resumen, el sistema operativo es una parte esencial del sistema informático y necesita tiempo de CPU para llevar a cabo sus funciones y proporcionar un entorno de ejecución eficiente para las aplicaciones.

2) Sí, en un sistema operativo (SO) pueden convivir procesos batch y procesos interactivos. Un SO moderno está diseñado para admitir una variedad de tipos de procesos y proporcionar recursos de manera equitativa y eficiente. Los procesos batch y los procesos interactivos pueden coexistir y ejecutarse en el mismo SO.

- Procesos Batch: Los procesos batch son programas que se ejecutan sin interacción del usuario y generalmente se inician automáticamente en un momento programado. Estos procesos a menudo se utilizan para realizar tareas repetitivas, como la generación de informes, procesamiento por lotes y otras operaciones que no requieren interacción en tiempo real.

- Procesos Interactivos: Los procesos interactivos son aquellos en los que los usuarios interactúan directamente con el sistema a través de una interfaz de usuario, como una línea de comandos o una interfaz gráfica. Estos procesos responden a las acciones del usuario en tiempo real y pueden incluir aplicaciones como navegadores web, editores de texto, reproductores multimedia, entre otros.

El SO administra todos los procesos de manera que puedan ejecutarse de manera coherente y justa, independientemente de si son procesos batch o interactivos. Utiliza políticas de planificación de procesos para asignar recursos de CPU y otros recursos de manera adecuada. Además, el SO puede ejecutar múltiples procesos concurrentemente utilizando técnicas como la multiprogramación y la multitarea.

En resumen, los sistemas operativos modernos están diseñados para admitir y gestionar procesos batch y procesos interactivos, permitiendo a los usuarios realizar tareas automatizadas y al mismo tiempo interactuar con el sistema de manera eficiente.

3) Sí, un sistema monousuario puede ser multitarea. La capacidad de ser multitarea no está limitada únicamente a sistemas multiusuario. La multitarea se refiere a la capacidad de ejecutar múltiples procesos o tareas de manera concurrente en un sistema, lo que puede mejorar la eficiencia y la utilización de recursos. Aunque la multitarea es común en sistemas multiusuario, también puede ser beneficiosa en sistemas monousuario.

Un sistema monousuario multitarea permite que un solo usuario realice varias tareas de manera concurrente. Por ejemplo, mientras un usuario está navegando por la web, también podría estar descargando archivos, ejecutando una aplicación de procesamiento de texto y escuchando música, todo al mismo tiempo. El sistema operativo se encarga de administrar la conmutación entre estos procesos de manera que parezca que se ejecutan simultáneamente.

La multitarea en un sistema monousuario puede mejorar la productividad y la experiencia del usuario al permitirle realizar múltiples tareas sin tener que esperar a que una tarea se complete antes de comenzar otra.

En resumen, un sistema monousuario puede ser multitarea si el sistema operativo que lo respalda admite la multitarea, lo que permite a un solo usuario ejecutar varios procesos o tareas concurrentemente.

4) Sí, un sistema multiusuario puede ser monotarea. La característica "multiusuario" se refiere a la capacidad de un sistema para admitir y gestionar múltiples cuentas de usuario, lo que permite que varios usuarios accedan y utilicen la misma computadora de manera concurrente. La característica "monotarea" se refiere a la capacidad de ejecutar una sola tarea o proceso a la vez.

Aunque la mayoría de los sistemas multiusuario son multitarea por naturaleza para permitir que varios usuarios realicen tareas simultáneamente, es posible que un sistema multiusuario esté configurado para que cada usuario tenga acceso exclusivo a la computadora de forma secuencial. En otras palabras, un usuario accedería y realizaría una tarea a la vez, y luego otro usuario podría acceder cuando el primero haya terminado. Esto sería un sistema multiusuario con capacidad monotarea.

Sin embargo, la capacidad multitarea suele ser una característica deseable en sistemas multiusuario para que varios usuarios puedan trabajar de manera eficiente y concurrente, y es común en la mayoría de los sistemas modernos. La capacidad de gestionar múltiples procesos de usuario y de sistema al mismo tiempo es una característica fundamental de la mayoría de los sistemas operativos multiusuario.

5) Sí, los programas pueden ejecutarse desde el disco en un sistema informático. En general, los programas y aplicaciones se almacenan en el disco duro o en otros dispositivos de almacenamiento, como unidades de estado sólido (SSD) o unidades flash USB, y se ejecutan desde allí. Cuando un usuario o el sistema operativo solicita la ejecución de un programa, el sistema operativo carga el programa en la memoria desde el disco antes de ejecutarlo. La ejecución de programas desde el disco es una parte fundamental del funcionamiento de un sistema informático.

6) Si, la memoria secundaria y la planificación de la CPU son dos aspectos diferentes de la gestión de recursos en un sistema informático. La memoria secundaria se utiliza para almacenar datos y programas que no caben en la memoria principal (RAM) del sistema, y se utiliza principalmente para mantener los datos persistentes, como archivos y aplicaciones en disco.

La planificación de la CPU se refiere a cómo se asigna y gestiona el tiempo de la CPU entre varios procesos que compiten por su tiempo de ejecución. No es necesario contar con memoria secundaria para realizar la planificación de la CPU. La planificación de la CPU se basa en la prioridad de los procesos, sus requisitos de tiempo de CPU y otros factores, y se realiza en la memoria principal (RAM).

En resumen, puedes planificar el uso de la CPU sin contar con memoria secundaria. Estos son aspectos independientes del funcionamiento del sistema operativo. La memoria secundaria es importante para el almacenamiento de datos y programas, pero no es un requisito para la planificación de la CPU.

7) No. En resumen, la interrupción por reloj es un mecanismo esencial para la gestión de la CPU y la ejecución de múltiples procesos en un sistema operativo multitarea. No evita que un proceso se apropie del procesador, pero contribuye a garantizar una asignación justa y eficiente de la CPU entre los procesos.

8) Si, Sí, las interrupciones son eventos externos a los procesos. Las interrupciones son señales generadas por hardware o software que interrumpen la ejecución normal de un proceso en una CPU. Estas interrupciones se utilizan para notificar al sistema operativo y a la CPU sobre eventos importantes o solicitudes de servicio que deben atenderse de inmediato.

Las interrupciones pueden ser de varios tipos, incluyendo:

1. Interrupciones de hardware: Generadas por dispositivos de hardware, como temporizadores, unidades de disco, teclados, controladores de red, etc. Estas interrupciones notifican al sistema operativo sobre eventos como datos listos para leer, errores de hardware, o solicitudes de entrada/salida.

2. Interrupciones de software: También conocidas como "llamadas al sistema", son generadas por programas o procesos que solicitan servicios del sistema operativo, como la creación de un nuevo proceso, la asignación de memoria, o la terminación de un proceso.

3. Interrupciones de reloj: La interrupción del reloj, como se mencionó anteriormente, es un tipo de interrupción generada periódicamente por un temporizador o reloj de hardware. Se utiliza para realizar tareas de planificación de procesos y programación de cambios de contexto.

Cuando ocurre una interrupción, el procesador detiene temporalmente la ejecución del proceso actual, guarda su estado actual y pasa a ejecutar un controlador de interrupciones correspondiente, que puede ser parte del sistema operativo. El controlador de interrupciones maneja la interrupción, realiza las tareas necesarias y luego permite que el proceso original continúe su ejecución. Esto permite al sistema operativo responder a eventos externos y garantizar la multitarea y la gestión eficiente de recursos.

9) No, un intento de acceder a una dirección ilegal generalmente se trata como una "violación de segmentación" o "segmentation fault" en sistemas Unix y Linux, y no se considera una "trap". Una violación de segmentación ocurre cuando un programa intenta acceder a una dirección de memoria a la que no tiene permiso o que no le corresponde, lo que generalmente indica un error en el código del programa.

Cuando se produce una violación de segmentación, el sistema operativo finaliza el proceso que la provocó y normalmente genera un volcado de núcleo (core dump) para ayudar a los desarrolladores a depurar el problema.

Por otro lado, un "trap" se refiere a una interrupción generada por una instrucción de software o hardware que puede ser manejada por el sistema operativo o el programa en ejecución. Las "traps" son un mecanismo que permite a un programa detectar y manejar eventos específicos, como divisiones por cero o acceso a una dirección de memoria nula, en lugar de causar una violación de segmentación.

En resumen, una violación de segmentación es un error que ocurre cuando un programa intenta acceder a una dirección de memoria no válida, mientras que un "trap" es un mecanismo para manejar eventos específicos y no se utiliza para describir errores de acceso a direcciones ilegales.

10) No, en un sistema operativo moderno y seguro, un proceso en modo usuario no puede acceder directamente al espacio de direcciones de memoria de otro proceso. Cada proceso se ejecuta en su propio espacio de direcciones de memoria aislado, lo que se conoce como espacio de direcciones virtual. Esto se hace por razones de seguridad y estabilidad del sistema.

El sistema operativo implementa protección de memoria y administración de memoria virtual para asegurarse de que un proceso no pueda acceder a la memoria de otros procesos. Si un proceso intenta acceder a la memoria de otro proceso, se genera una excepción de protección de memoria (como una violación de segmentación en sistemas Unix y Linux) y el sistema operativo lo detiene, evitando así el acceso no autorizado a la memoria de otros procesos.

Es importante destacar que el acceso entre procesos generalmente se logra a través de mecanismos de comunicación interproceso (IPC, por sus siglas en inglés) proporcionados por el sistema operativo, como tuberías (pipes), sockets, señales, colas de mensajes, entre otros. Estos mecanismos permiten a los procesos compartir información de manera controlada y segura.

11. No, una llamada al sistema (system call) no genera necesariamente la creación de un proceso del sistema operativo. Las llamadas al sistema son utilizadas por los programas de aplicación para solicitar servicios o recursos del sistema operativo, como lectura/escritura de archivos, administración de procesos, comunicación entre procesos, etc. El sistema operativo maneja estas solicitudes y proporciona los servicios solicitados sin necesidad de crear un nuevo proceso por cada llamada al sistema.

13. No necesariamente. La disponibilidad de procesos orientados a entrada/salida y el estado de las colas de solicitudes a los dispositivos de E/S dependen de la carga y la actividad del sistema en un momento dado. Si tienes muchos procesos esperando realizar operaciones de entrada/salida, las colas de solicitudes a los dispositivos de E/S podrían estar ocupadas. Sin embargo, no se puede afirmar que siempre estarán vacías, ya que esto dependerá de la carga de trabajo del sistema y de la cantidad de solicitudes de E/S que se realicen en un momento específico.

Continuemos con tus preguntas:

14. Sí, uno de los propósitos fundamentales del sistema operativo es permitir que los usuarios y los programas de aplicación se abstraigan del hardware subyacente y su manejo. El sistema operativo proporciona una interfaz de alto nivel que oculta la complejidad del hardware, permitiendo a los usuarios y programas interactuar con el sistema de una manera más sencilla y consistente.

15. No, el kernel no es lo mismo que el sistema operativo en su totalidad. El kernel es una parte esencial del sistema operativo que se encarga de la administración de recursos, la programación de procesos y otras tareas críticas. Sin embargo, el sistema operativo incluye además bibliotecas, utilidades, aplicaciones y otros componentes que permiten a los usuarios interactuar con la computadora. El kernel es un componente central, pero no es todo el sistema operativo.

16. La memoria principal, también conocida como RAM (Random Access Memory), no es un recurso multiplexado en el espacio. En cambio, es un recurso compartido de acceso aleatorio al que pueden acceder directamente tanto el kernel como los procesos en ejecución. La memoria RAM es un recurso crítico para el sistema operativo y los programas, y se utiliza para almacenar datos y programas en ejecución. A diferencia de los dispositivos de E/S, que pueden ser multiplexados, la memoria RAM se utiliza para almacenar datos en tiempo real y se accede de forma rápida y directa.

Continuemos con tus preguntas:

17. Sí, en un sistema monoprocesador, el procesador es un recurso multiplexado en el tiempo a través de la conmutación de procesos. Aunque solo hay un procesador físico en este tipo de sistemas, el sistema operativo administra varios procesos y asigna el tiempo de CPU a cada proceso en intervalos pequeños. Esto crea la ilusión de que múltiples procesos se ejecutan simultáneamente, pero en realidad, el procesador conmuta entre ellos rápidamente.

18. Sí, `date` es un comando que generalmente se implementa mediante una llamada al sistema (system call). La llamada al sistema correspondiente permite a un programa obtener la fecha y la hora actual del sistema.

19. Verdadero. En muchos sistemas, un proceso tiene dos pilas (stacks): uno en modo usuario y otro en modo supervisor. Sin embargo, estos stacks no ocupan la misma dirección de memoria al mismo tiempo. Se utilizan en momentos diferentes: el stack en modo usuario se usa cuando el proceso está en modo usuario, y el stack en modo supervisor se utiliza cuando el proceso pasa al modo supervisor (kernel mode). Cuando se cambia entre el modo usuario y el modo supervisor, el hardware se encarga de cambiar las referencias de pila, de modo que no ocupan la misma dirección de memoria al mismo tiempo. Esto es fundamental para la seguridad y el aislamiento entre los procesos y el kernel del sistema operativo.

Sigamos con tus preguntas:

20. Verdadero (V). El estado del proceso se almacena en la PCB (Bloque de Control de Proceso), que es una estructura de datos en el sistema operativo que contiene información sobre el estado del proceso, como los registros del procesador, el contador de programa, las prioridades y otros detalles relacionados con la ejecución del proceso.

21. Verdadero (V). Un proceso puede crear a otro proceso mediante una llamada al sistema específica, como `fork()` en sistemas basados en Unix. La llamada al sistema `fork()` crea un nuevo proceso que es una copia del proceso padre. Esta es una forma común de crear procesos secundarios en sistemas operativos.

22. Falso (F). La cola de procesos, o la cola de listos, no se almacena en el disco. Normalmente, la cola de procesos listos se mantiene en la memoria principal, en una estructura de datos gestionada por el sistema operativo. Los procesos listos esperan su turno en esta cola para ser ejecutados por el procesador.

23. Verdadero (V). Cuando se crea un proceso, generalmente se encuentra en estado de espera en disco hasta que el sistema operativo le asigna recursos y tiempo de CPU para ejecutarse. Los programas o ejecutables en disco no se ejecutan directamente desde allí; primero se crean procesos en memoria a partir de estos programas.

24. Falso (F). El proceso hijo se crea en su propio espacio de direcciones, que es una copia del espacio de direcciones del proceso padre. Esto se hace para mantener el aislamiento entre los procesos y evitar que un proceso afecte directamente el espacio de direcciones de otro proceso.

25. Verdadero (V). Las tablas de archivos correspondientes a los archivos abiertos por un proceso forman parte de su contexto. Estas tablas contienen información sobre los archivos abiertos, como descriptores de archivos, posición actual en el archivo, modos de acceso, entre otros. Mantener esta información en el contexto del proceso es esencial para que el proceso pueda realizar operaciones de E/S (entrada/salida) en los archivos abiertos.

Sigamos con las respuestas:

26. Verdadero (V). La PCB (Bloque de Control de Proceso) se crea cuando un proceso se carga en la memoria principal. La PCB contiene información relevante sobre el proceso y se inicializa cuando el proceso se carga en memoria.

27. Falso (F). Después de una llamada al sistema `fork()`, el proceso padre y el proceso hijo tienen copias separadas de la PCB. Cada proceso tiene su propia PCB con información independiente sobre su estado, registros y otros detalles. La copia de la PCB en el proceso hijo se inicializa de acuerdo con la información del proceso padre.

28. Verdadero (V). Si no hubiera operaciones de E/S, los procesos no necesitarían realizar llamadas al sistema para interactuar con dispositivos de E/S, como lectura o escritura en archivos. Las llamadas al sistema son necesarias para permitir que los procesos interactúen con el sistema operativo y realicen operaciones que requieran privilegios especiales, como acceso a archivos o administración de recursos.

29. Falso (F). En modo supervisor (modo kernel), no es posible acceder directamente al espacio de direcciones de otros procesos. El sistema operativo garantiza el aislamiento entre los procesos al no permitir que un proceso acceda al espacio de direcciones de otro proceso. Esto se hace para garantizar la seguridad y la protección de la integridad de los procesos.

30. Falso (F). El contexto de un proceso es diferente de su espacio de direcciones. El contexto de un proceso incluye información sobre el estado del proceso, registros, prioridad, propiedades de E/S, entre otros detalles relacionados con la ejecución del proceso. El espacio de direcciones se refiere a la memoria asignada al proceso para almacenar su código, datos y pila. Ambos conceptos son distintos pero importantes para el funcionamiento de un proceso en un sistema operativo.

Continuemos con las respuestas:

31. b) cuanto tiempo de espera tiene acumulado. La prioridad dinámica o aging por inanición se basa en dar prioridad a los procesos que han esperado mucho tiempo en la cola de listos. Esto se logra acumulando tiempo de espera para cada proceso y aumentando su prioridad con el tiempo.

32. Verdadero (V). Un cambio de modo implica un cambio de contexto. Cuando un proceso pasa de ejecución en modo usuario a modo supervisor (kernel) o viceversa, se realiza un cambio de contexto para asegurar que el proceso pueda ejecutar código privilegiado o regresar al modo usuario.

33. Verdadero (V). Un cambio de contexto involucra un cambio de modo. Cuando se produce un cambio de contexto, el procesador pasa de ejecutar un proceso en modo usuario a ejecutar el núcleo del sistema operativo en modo supervisor (kernel) o viceversa. Esto es necesario para realizar operaciones que requieran privilegios especiales.

34. Falso (F). Un cambio de contexto no es lo mismo que un cambio de proceso. Un cambio de contexto se refiere a la transición del procesador entre la ejecución de diferentes procesos, mientras que un cambio de proceso implica que un proceso específico se reemplaza por otro en la CPU. Un cambio de contexto puede ocurrir dentro de un proceso en sistemas con multitarea cooperativa.

35. Verdadero (V). Un cambio de contexto implica un cambio de modo, ya que el procesador cambia entre la ejecución en modo usuario y modo supervisor (kernel). El cambio de modo es necesario para permitir que el sistema operativo realice operaciones privilegiadas y, a su vez, para garantizar la protección del sistema operativo y la seguridad del sistema.

Continuemos con las respuestas:

36. Un fork exitoso produce cambios en la PCB del padre, pues se almacena el PID (identificador de proceso) del hijo. El padre necesita conocer el PID del hijo para gestionar y comunicarse con él.

37. En el mecanismo de manejo de memoria con particiones, el espacio de direcciones de un proceso está delimitado por los registros "base" y "límite." Estos registros definen la ubicación y el tamaño de la partición de memoria asignada al proceso.

38. Un fork devuelve dos valores: 0 al proceso hijo y el PID del hijo al proceso padre. El valor 0 es utilizado por el proceso hijo para identificarse como el proceso recién creado, y el PID del hijo es necesario para que el proceso padre pueda gestionar al nuevo hijo.

39. Un acceso no autorizado a memoria es detectado por el Hardware. El hardware del sistema puede detectar intentos de acceso a áreas de memoria protegidas y generar una excepción o interrupción para que el sistema operativo tome medidas adecuadas.

40. Verdadero (V). Las System Calls se ejecutan en "Modo Privilegiado" (modo supervisor o kernel mode) para que puedan realizar operaciones que requieran privilegios especiales, como acceder a recursos del sistema, realizar llamadas al sistema, gestionar la memoria, etc. Esto garantiza la seguridad y la integridad del sistema.

Continuemos con las respuestas:

41. Ante un cambio de contexto, en la PCB se guarda:

a) La tabla de páginas: para gestionar la memoria virtual del proceso.

b) La pila de usuario: para restaurar el estado de la pila del proceso.

c) La tabla de archivos abiertos: para mantener la información sobre los archivos abiertos por el proceso.

d) El estado del proceso: que incluye información como los registros de la CPU, el contador de programa, el estado del proceso (listo, en ejecución, bloqueado, etc.), el PID, entre otros datos relevantes.

42. No, el chequeo de la existencia de una interrupción no se realiza entre los pasos de "Fetch" y "Execute" de cada ciclo de instrucción. El "Fetch" y "Execute" son pasos del ciclo de instrucción de una CPU en la ejecución de un programa. El chequeo de interrupciones se realiza a nivel del hardware de la CPU y generalmente ocurre antes o después de completar una instrucción. Cuando se detecta una interrupción, el flujo de control se desvía al manejador de interrupciones correspondiente, lo que puede resultar en un cambio de contexto y una respuesta a la interrupción, como la ejecución de un servicio del sistema o la conmutación de procesos.

Continuemos con las respuestas:

43. Sí, el vector de interrupciones generalmente se encuentra en memoria. Este vector es una tabla que contiene las direcciones de los manejadores de interrupciones para diferentes tipos de interrupciones. Cuando ocurre una interrupción, el sistema operativo busca en el vector de interrupciones la dirección del manejador correspondiente en memoria para atender la interrupción.

44. No necesariamente. La llamada al sistema `fork` crea un nuevo proceso hijo como una copia exacta del proceso padre. Aunque se crea un nuevo proceso, no necesariamente se produce un cambio de contexto inmediato. El cambio de contexto depende de la planificación del sistema operativo y de si el nuevo proceso hijo se programa para ejecutarse inmediatamente o después de algún evento de conmutación de procesos.

45. No, un proceso swappeado en estado listo (ready to run) aún compite por la CPU una vez que se le devuelve a la memoria principal. Cuando un proceso es swappeado, se encuentra en estado suspendido o en la memoria secundaria. Cuando se le vuelve a cargar en la memoria principal y se encuentra en estado listo, está listo para competir por la CPU.

46. No necesariamente. El scheduler de short term (o planificación de corto plazo) generalmente se ejecuta con mayor frecuencia que el de long term (planificación de largo plazo). El scheduler de corto plazo se encarga de decidir cuál de los procesos listos en memoria se ejecutará a continuación, mientras que el scheduler de largo plazo toma decisiones menos frecuentes sobre cuándo cargar nuevos procesos en memoria.

47. No, el cambio de contexto generalmente es realizado por el scheduler de short term (planificación de corto plazo). Este scheduler decide cuál de los procesos listos en memoria se ejecutará a continuación y realiza el cambio de contexto al cargar el proceso elegido en la CPU.

48. No necesariamente. Cuando a un proceso se le termina su quantum, generalmente pasa de vuelta al estado "listo" en lugar de "espera". En el estado "listo", el proceso está listo para competir nuevamente por la CPU. El cambio de contexto puede ocurrir si el scheduler de corto plazo decide ejecutar otro proceso en lugar del que agotó su quantum.

Continuemos con las respuestas:

49. Falso. El scheduler de medium term (planificación de medio plazo) no decide el cambio entre los estados "nuevo" y "listo". El cambio entre estos estados se produce principalmente debido a la llegada de nuevos procesos a la cola de listos. El scheduler de medio plazo suele estar relacionado con la gestión de procesos en memoria, como la decisión de cargar o expulsar procesos desde la memoria principal a la memoria secundaria.

50. Verdadero. El scheduler de short term (planificación de corto plazo) es el encargado de seleccionar qué proceso de la cola de listos (ready) se ejecutará a continuación y hace que el proceso pase del estado "listo" al estado "running" (ejecución en CPU).

51. Verdadero. En la planificación de CPU, el objetivo principal es maximizar la productividad y minimizar el tiempo de respuesta de los procesos. Se busca que la CPU se utilice eficazmente y que los procesos se ejecuten de manera eficiente.

52. Verdadero. El tiempo de retorno de un proceso es el tiempo transcurrido desde que se inicia su ejecución hasta que finaliza, y generalmente incluye el tiempo que pasa en la CPU y el tiempo de espera en colas, si lo hay. Es una medida importante para evaluar el rendimiento de un proceso y un sistema de planificación.

53. Falso. Cuando un proceso está en espera y se cumple el evento por el que estaba esperando, generalmente pasa al estado "listo" en memoria principal en lugar de en memoria secundaria. El estado "listo" significa que el proceso está listo para competir por la CPU una vez que le llegue su turno en la cola de listos. El paso a memoria secundaria se refiere a la suspensión del proceso en disco cuando no se encuentra en condiciones de ejecución y se requiere espacio en memoria principal para otros procesos en ejecución.

55. Sí, un proceso puede pasar de esperar en memoria secundaria a esperar en memoria principal cuando se intercambia entre memoria secundaria y memoria principal según la necesidad.

56. Verdadero, el scheduler de medium term maneja el grado de multiprogramación al decidir cuántos procesos pueden mantenerse en memoria en un momento dado.

57. Sí, el uso del disco permitió implementar la planificación de procesos al proporcionar un mecanismo para almacenar y recuperar procesos de memoria principal a memoria secundaria.

58. Falso, en un sistema monoprocesador, cuando se atiende una interrupción, el proceso en ejecución puede ser pausado temporalmente para manejar la interrupción, pero no todos los procesos quedan en espera.

59. Sí, en un ambiente con procesos interactivos y batch, puede ser beneficioso utilizar algoritmos apropiativos para dar prioridad a los procesos interactivos y proporcionar una respuesta más rápida a las solicitudes del usuario.

60. La secuencia de actividades que realiza el dispatcher suele ser:

a) Cambio de contexto

c) Salto a la primera/próxima instrucción a ejecutar

El dispatcher es responsable de realizar un cambio de contexto para cambiar de un proceso a otro, lo que implica guardar el estado del proceso actual y cargar el estado del nuevo proceso seleccionado en la CPU. Luego, salta a la primera instrucción del nuevo proceso para continuar su ejecución.

61. b) Se saturan las colas de dispositivo.

Cuando se tienen muchos procesos orientados a E/S, pueden llenarse las colas de dispositivos debido a la alta demanda de E/S por parte de los procesos, lo que puede resultar en una saturación de las colas de dispositivo.

62. Falso, cuando se carga un proceso en memoria, se hace en modo supervisor (kernel mode) para acceder a la tabla de páginas y asignar la memoria física adecuada al proceso.

63. Falso, la dirección que se carga en el PC (Program Counter) es una dirección virtual (lógica), no una dirección física. La traducción de la dirección lógica a dirección física se realiza mediante la unidad de gestión de memoria.

64. Falso, en las particiones dinámicas, la opción "worst fit" tiende a ser menos eficiente que otras estrategias de asignación de particiones, como "best fit" o "first fit," ya que puede dejar fragmentación externa significativa.

65. La resolución de una dirección que involucra el contenido del registro de reubicación (base) y una dirección lógica se realiza mediante la unidad de gestión de memoria, específicamente, el hardware de traducción de direcciones (MMU). La MMU se encarga de convertir la dirección lógica en una dirección física para acceder a la memoria principal.

66. Ventajas de la paginación pura en comparación con cargar todo el proceso en memoria de forma contigua:

Ventajas:

- Mayor utilización de la memoria: En la paginación pura, solo se cargan en memoria las páginas que se utilizan actualmente, lo que permite un uso más eficiente de la memoria, especialmente cuando los procesos son grandes y no se necesita cargar todo el proceso en memoria de una vez.

- Mayor flexibilidad: La paginación pura permite a los procesos crecer o reducir su espacio de direcciones virtual de manera más flexible, ya que no requiere cargar el proceso de manera contigua.

- Facilita la administración de la memoria: La gestión de la memoria es más sencilla en un sistema de paginación pura, ya que no se requiere una asignación contigua de memoria física.

Desventajas:

- Sobrecarga por las tablas de páginas: La paginación pura puede requerir tablas de páginas más grandes, lo que implica una sobrecarga adicional en términos de espacio y tiempo.

- Fragmentación interna: Puede haber fragmentación interna en las páginas, lo que significa que algunas páginas pueden no utilizarse por completo.

- Mayor latencia: La traducción de direcciones lógicas a direcciones físicas a través de las tablas de páginas puede introducir cierta latencia en el acceso a la memoria.

67. Verdadero, en la paginación, los procesos utilizan direcciones lógicas que deben traducirse en direcciones físicas mediante las tablas de páginas.

68. En la paginación pura, es necesario guardar en la entrada de la tabla de páginas:

- El número de página (Page Number): Para indicar a qué página del proceso se refiere la entrada.

- El número de marco de página (Frame Number): Para indicar la ubicación de la página en la memoria física.

Estas dos piezas de información son necesarias para mapear direcciones lógicas a direcciones físicas y permitir el acceso a las páginas del proceso.