1. ¿El SO necesita tiempo de CPU?

Sí, el sistema operativo (SO) necesita tiempo de CPU para funcionar correctamente. El sistema operativo es un software fundamental que administra y controla todos los recursos de hardware de un sistema de computadora, incluyendo la CPU. El SO realiza una variedad de tareas esenciales, como la gestión de procesos, la administración de memoria, la administración de dispositivos de entrada/salida, la programación de la CPU, la gestión de archivos, la seguridad y muchas otras funciones críticas.

Para llevar a cabo todas estas tareas, el sistema operativo debe ejecutarse en la CPU. El SO tiene su propio conjunto de procesos y subprocesos que se ejecutan en la CPU para realizar estas funciones de administración y control. Por lo tanto, el tiempo de CPU se asigna al sistema operativo para permitir que realice sus tareas de gestión y para garantizar el funcionamiento adecuado del sistema en su conjunto.

En un sistema multiproceso o multitarea, el SO también es responsable de la planificación de la CPU y la asignación de tiempo de CPU a los procesos de usuario y al propio sistema operativo. Esto se hace mediante algoritmos de planificación de la CPU que garantizan que el SO tenga la oportunidad de ejecutar sus tareas de forma justa y eficiente.

En resumen, el sistema operativo requiere tiempo de CPU para llevar a cabo sus funciones de administración y control, lo que es fundamental para el funcionamiento de una computadora o sistema informático.

2. ¿Pueden convivir en un mismo SO procesos batch y procesos interactivos?

Sí, en un sistema operativo (SO) pueden convivir procesos batch y procesos interactivos. Los SO modernos están diseñados para admitir una amplia variedad de tipos de procesos, incluidos los procesos batch y los procesos interactivos, y proporcionan mecanismos para gestionarlos de manera eficiente.

Procesos Batch:

Los procesos batch son programas que se ejecutan sin interacción directa del usuario. Estos procesos a menudo se utilizan para tareas que requieren mucho tiempo de CPU, como procesamiento por lotes de datos, generación de informes, procesamiento de archivos por lotes, etc.

En un SO, los procesos batch pueden ejecutarse en segundo plano sin requerir interacción del usuario. El SO puede programar estos procesos para que se ejecuten en momentos específicos o cuando los recursos de la CPU estén disponibles.

Procesos Interactivos:

Los procesos interactivos son aquellos que interactúan directamente con el usuario a través de una interfaz de usuario, como una línea de comandos o una interfaz gráfica de usuario (GUI). Estos procesos incluyen aplicaciones de usuario, como navegadores web, editores de texto, reproductores multimedia, etc.

Los procesos interactivos requieren una respuesta rápida a las acciones del usuario y, por lo tanto, se les da prioridad en la asignación de tiempo de CPU.

El SO gestiona tanto procesos batch como interactivos utilizando algoritmos de planificación de la CPU que equilibran las necesidades de ambos tipos de procesos. Los procesos interactivos a menudo reciben un mayor grado de prioridad para garantizar una experiencia de usuario receptiva, mientras que los procesos batch pueden programarse para ejecutarse en momentos en que la carga del sistema sea más ligera.

En resumen, los procesos batch y procesos interactivos pueden coexistir en un mismo SO, y el sistema operativo se encarga de administrarlos de manera eficiente para satisfacer las necesidades de ambos tipos de procesos.

3. ¿Puede un sistema monousuario ser multitarea?

Sí, un sistema monousuario puede ser multitarea, es decir, puede admitir la ejecución simultánea de múltiples tareas o procesos en un entorno en el que solo un usuario interactúa con el sistema a la vez. La multitarea en un sistema monousuario se refiere a la capacidad del sistema operativo para administrar y ejecutar múltiples procesos o aplicaciones al mismo tiempo, lo que permite que el usuario realice varias tareas de manera eficiente.

A pesar de que solo un usuario interactúa con el sistema a la vez, la multitarea en un sistema monousuario ofrece ventajas como la capacidad de ejecutar múltiples aplicaciones de forma concurrente. Algunos ejemplos de situaciones en las que la multitarea en un sistema monousuario puede ser útil incluyen:

Ejecutar varios programas al mismo tiempo: Un usuario puede, por ejemplo, ejecutar un procesador de texto mientras escucha música, navega por la web o realiza otras tareas en segundo plano.

Administrar tareas en segundo plano: Un sistema monousuario puede realizar tareas en segundo plano, como actualizaciones automáticas, copias de seguridad o indexación de archivos, mientras el usuario trabaja con una aplicación principal.

Mejorar la eficiencia: La multitarea permite al usuario alternar rápidamente entre aplicaciones y tareas, lo que mejora la eficiencia y la productividad.

El sistema operativo en un sistema monousuario aún debe administrar los recursos de hardware y garantizar que los procesos se ejecuten de manera justa y eficiente. Los sistemas operativos modernos, incluso en entornos monousuario, son capaces de gestionar múltiples tareas de manera eficiente y proporcionar una experiencia de usuario multitarea.

Por lo tanto, aunque solo un usuario interactúe con el sistema, un sistema monousuario puede ser perfectamente capaz de admitir la multitarea, lo que facilita la realización de múltiples tareas y la gestión eficiente de aplicaciones y procesos.

4. ¿Puede un sistema multiusuario ser monotarea?

Sí, es posible que un sistema multiusuario sea monotarea, lo que significa que solo permite que un usuario realice una tarea o ejecute un proceso a la vez. Aunque los sistemas multiusuario generalmente están diseñados para admitir múltiples usuarios concurrentes y ejecutar múltiples procesos al mismo tiempo, la capacidad de multitarea puede ser deshabilitada o limitada por razones de seguridad, rendimiento o políticas del sistema.

Algunas razones por las que un sistema multiusuario podría ser monotarea incluyen:

Seguridad: En entornos de seguridad crítica o en situaciones donde se busca minimizar el riesgo de interferencia entre usuarios, un sistema puede restringir la multitarea para garantizar que un usuario no pueda interferir con el trabajo de otros.

Recursos Limitados: En sistemas con recursos limitados, como sistemas embebidos o sistemas más antiguos, la multitarea puede ser deshabilitada para asegurar que los recursos se asignen completamente a la tarea o proceso actual, evitando la sobrecarga del cambio de contexto entre procesos.

Requerimientos Específicos: Algunas aplicaciones o tareas críticas pueden requerir acceso exclusivo a los recursos del sistema, lo que puede limitar la capacidad de multitarea.

Políticas del Sistema: Las políticas de administración de sistemas pueden imponer restricciones en la multitarea según las necesidades específicas de la organización.

En resumen, aunque la mayoría de los sistemas multiusuario están diseñados para admitir la multitarea, es posible limitar o desactivar esta capacidad en situaciones especiales. La capacidad de multitarea en un sistema multiusuario dependerá de la configuración y las políticas específicas del sistema, así como de las necesidades de los usuarios y la seguridad.

5. ¿Puede un programa ejecutarse desde el disco?

Si se puede... pero es muy lento, lo normal es ejecutarlos una vez se cargaron en la ram

Por lo visto en la materia creo que no xd

6. ¿Puedo planificar el uso de la CPU si no cuento con memoria secundaria?

En resumen, la planificación de la CPU se puede realizar sin memoria secundaria. La memoria principal es fundamental para la ejecución de procesos y la gestión de recursos de la CPU, y la memoria secundaria es útil para el almacenamiento a largo plazo y la gestión de datos, pero no es necesaria para la planificación de la CPU en sí.

osea puedo prescindir de memoria secundaria, pero no de memoria principal(ram)

NO, porque ahí se guardan los programas que voy a usar

7. La interrupción por clock impide que un proceso se apropie del procesador.

Por lo visto en la materia si...

La interrupción por reloj no impide que un proceso se apropie permanentemente del procesador, pero sí juega un papel importante en evitar que un proceso se adueñe de la CPU de manera indefinida. Esta interrupción se utiliza para programar cambios de contexto y realizar la planificación de la CPU en sistemas multitarea o multiproceso.

Cuando ocurre una interrupción por reloj, el sistema operativo tiene la oportunidad de tomar el control del procesador y decidir si debe permitir que el proceso actual continúe ejecutándose o cambiar a otro proceso. Los sistemas operativos suelen utilizar algoritmos de planificación de la CPU, como el planificador de round-robin, que asigna un tiempo de ejecución limitado (quantum) a cada proceso. Cuando se agota el tiempo del quantum o cuando ocurre una interrupción por reloj, el sistema operativo cambia al siguiente proceso en la cola de tareas pendientes.

Este enfoque asegura que ningún proceso monopolice permanentemente la CPU, ya que se le asigna un tiempo limitado y, después de eso, otros procesos tienen la oportunidad de ejecutarse. Sin embargo, es importante destacar que si un proceso está diseñado para usar todos los recursos disponibles de manera intensiva o está programado de manera ineficiente, aún puede ocupar gran parte del tiempo de CPU durante su turno.

En resumen, la interrupción por reloj y la planificación de la CPU evitan que un proceso se adueñe permanentemente del procesador, pero no lo impiden por completo. Ayudan a garantizar una asignación justa y eficiente del tiempo de CPU entre múltiples procesos para mantener el sistema receptivo y equitativo.

8. Las interrupciones son externas a los procesos.

Sí, las interrupciones son eventos externos a los procesos que ocurren en un sistema informático y que pueden provocar una interrupción en la ejecución normal de un proceso en curso. Las interrupciones son mecanismos importantes para la gestión de recursos y el control de un sistema. Aquí hay algunos ejemplos de interrupciones:

Interrupción de hardware: Estas interrupciones se generan por eventos físicos en el hardware, como un error de hardware, una señal de temporizador (interrupción por reloj), una solicitud de E/S (entrada/salida), o una pulsación de tecla en un teclado. Cuando se produce una interrupción de hardware, el procesador puede suspender temporalmente la ejecución del proceso actual y pasar a manejar la interrupción.

Interrupción de software: Estas interrupciones son generadas por instrucciones de software, como llamadas al sistema operativo (system calls) o excepciones de software, que requieren la atención del sistema operativo. Por ejemplo, un programa puede solicitar servicios del sistema operativo, como abrir un archivo o asignar memoria, a través de una llamada al sistema, lo que desencadena una interrupción de software.

Interrupciones de reloj: Las interrupciones por reloj son generadas periódicamente por el reloj del sistema y se utilizan para realizar la planificación de la CPU, como se mencionó en respuestas anteriores.

Interrupciones externas: Estas interrupciones pueden provenir de fuentes externas, como dispositivos periféricos o hardware adicional conectado al sistema. Un ejemplo común es la interrupción generada por un mouse o un controlador de red cuando se produce una interacción con el usuario o cuando llegan datos desde una red.

En todos estos casos, las interrupciones permiten que el sistema maneje eventos externos o excepciones de manera efectiva sin que los procesos en ejecución sean responsables de gestionarlos directamente. El sistema operativo o el controlador de interrupciones del sistema se encarga de manejar y dirigir las interrupciones de manera adecuada.

9. Un intento de acceder a una dirección ilegal, se trata como un trap.

Sii, porque es algo INTERNO al proceso

Un intento de acceso a una dirección ilegal o una instrucción no válida se trata generalmente como una excepción o una interrupción de software en lugar de un "trap". En el contexto de sistemas informáticos, un "trap" se refiere a una instrucción o evento programado que provoca una transición a un modo privilegiado o al kernel del sistema operativo.

Cuando un programa intenta acceder a una dirección de memoria que no está asignada o utiliza una instrucción no válida, se genera una excepción de hardware o una excepción de software, dependiendo de la arquitectura del sistema y el diseño del hardware. La gestión de excepciones y errores de hardware suele ser responsabilidad del sistema operativo y del controlador de interrupciones del sistema.

En el caso de una excepción, el sistema operativo o el controlador de interrupciones se encargan de manejar la situación. Pueden tomar medidas como finalizar el proceso que causó la excepción o generar un error en el sistema. En algunos casos, el sistema operativo puede registrar información sobre la excepción para el diagnóstico y la depuración.

La distinción entre "trap" y "excepción" puede variar según la terminología utilizada en diferentes sistemas o arquitecturas. En general, un "trap" se considera una instrucción específica utilizada para generar una interrupción de software en el sistema, mientras que una excepción se refiere a situaciones no esperadas o errores en la ejecución de un programa.

10. Un proceso puede acceder al espacio de direcciones de otro proceso si esta en modo usuario.

En resumen, en un sistema operativo moderno, un proceso en modo usuario no puede acceder directamente al espacio de direcciones de otro proceso. El sistema operativo actúa como intermediario para administrar el acceso y garantizar la seguridad y el aislamiento entre procesos.

Esto prodia pasar solo en modo kernel

11. Una llamada al sistema (system call) genera la creación de un proceso del sistema operativo para atender la llamada.

no

No, una llamada al sistema (system call) no genera la creación de un proceso del sistema operativo para atender la llamada en sí. En cambio, una llamada al sistema es un mecanismo que permite a los procesos en modo usuario solicitar servicios o funcionalidades proporcionadas por el sistema operativo.

Cuando un proceso en modo usuario realiza una llamada al sistema, lo que ocurre es lo siguiente:

El proceso en modo usuario solicita un servicio específico o una operación al sistema operativo, como abrir un archivo, crear un proceso hijo, realizar una operación de E/S, entre otros.

El sistema operativo recibe la solicitud y, en lugar de crear un proceso del sistema operativo, pasa a un modo privilegiado, que es el modo kernel o modo supervisor, donde tiene acceso completo a los recursos del sistema.

El sistema operativo ejecuta la operación o servicio solicitado en nombre del proceso en modo usuario, utilizando sus propios recursos y privilegios.

Una vez que se completa la operación, el sistema operativo devuelve el control al proceso en modo usuario, que continúa su ejecución.

La llamada al sistema permite que los procesos en modo usuario accedan a funciones y servicios proporcionados por el sistema operativo de una manera controlada y segura, sin necesidad de crear un proceso separado para atender cada llamada al sistema. El sistema operativo es el encargado de administrar las llamadas al sistema y ejecutar las operaciones solicitadas de manera eficiente y segura.

12. Las llamadas al sistema son la forma que tienen las aplicaciones de comunicarse con el sistema operativo.

Así es, las llamadas al sistema (system calls) son la forma que tienen las aplicaciones de comunicarse con el sistema operativo en la mayoría de los sistemas operativos. Las llamadas al sistema son interfaces definidas que permiten a los programas en modo usuario solicitar servicios y funcionalidades proporcionadas por el sistema operativo. Estas llamadas son esenciales para que las aplicaciones interactúen con el hardware y realicen tareas como la gestión de archivos, la comunicación en red, la asignación de memoria, la administración de procesos y más.

Algunos ejemplos de operaciones que se pueden realizar mediante llamadas al sistema incluyen:

Abrir, leer, escribir y cerrar archivos.

Asignar y liberar memoria.

Crear, gestionar y finalizar procesos.

Realizar operaciones de entrada/salida (E/S).

Establecer conexiones de red y comunicarse a través de sockets.

Acceder a información del sistema, como la hora actual o la configuración del sistema.

Gestionar señales y manejar eventos del sistema.

Las llamadas al sistema proporcionan una interfaz estandarizada y segura para que las aplicaciones interactúen con el núcleo del sistema operativo. Estas llamadas son gestionadas por el sistema operativo, que verifica los permisos y la seguridad de cada solicitud para garantizar un funcionamiento adecuado y evitar posibles problemas de seguridad.

En resumen, las llamadas al sistema son esenciales para que las aplicaciones se comuniquen con el sistema operativo y realicen diversas tareas, lo que permite a las aplicaciones aprovechar la funcionalidad proporcionada por el sistema operativo de manera controlada y segura.

13. Si tengo muchos procesos orientados a entrada/salida, las colas de solicitudes a los dispositivos de E/S estarán vacías.

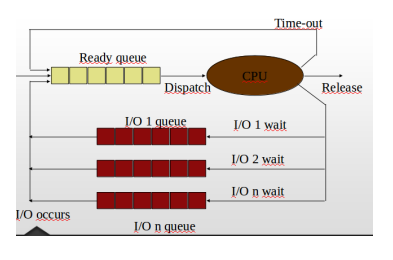
Mm no, no estaran vacias

Correcto, si tienes muchos procesos orientados a entrada/salida (I/O-bound), es probable que las colas de solicitudes a los dispositivos de E/S no estén vacías en un sistema. Los procesos I/O-bound tienden a realizar operaciones de entrada/salida (lecturas, escrituras, comunicaciones en red, etc.), y estas operaciones a menudo involucran esperas significativas debido a la latencia de los dispositivos.

Cuando un proceso I/O-bound inicia una operación de E/S, puede enviar una solicitud al dispositivo de E/S correspondiente y luego liberar la CPU mientras espera a que se complete la operación. Mientras tanto, las solicitudes de E/S pendientes se mantienen en las colas de solicitudes a los dispositivos de E/S. Estas colas se gestionan y administran por el sistema operativo y los controladores de dispositivos.

La gestión eficiente de las colas de solicitudes a dispositivos de E/S es esencial para garantizar que las operaciones de E/S se realicen de manera eficiente y que los procesos I/O-bound puedan continuar funcionando sin bloquear la CPU en espera activa.

En resumen, en sistemas con muchos procesos orientados a entrada/salida, las colas de solicitudes a dispositivos de E/S a menudo contendrán solicitudes pendientes, ya que los procesos I/O-bound dependen de operaciones de E/S para realizar su trabajo y, por lo tanto, generan solicitudes de manera regular. La gestión adecuada de estas colas es una parte fundamental de la administración de recursos en el sistema operativo.



14. ¿EI sistema operativo permite al usuario abstraerse del hardware y su manejo.

Sí, uno de los roles fundamentales de un sistema operativo es permitir al usuario abstraerse del hardware y su manejo. El sistema operativo actúa como una capa intermedia entre el hardware subyacente y las aplicaciones de usuario, proporcionando una interfaz de usuario más amigable y abstracta que facilita el uso de la computadora.

Aquí hay algunas formas en las que el sistema operativo facilita la abstracción del hardware:

Gestión de recursos: El sistema operativo administra eficientemente los recursos de hardware, como la CPU, la memoria, el almacenamiento y los dispositivos de E/S, lo que permite a las aplicaciones acceder a estos recursos sin necesidad de conocer los detalles específicos del hardware subyacente.

Interfaz de usuario: El sistema operativo proporciona una interfaz de usuario que simplifica la interacción con la computadora. Esto puede incluir interfaces gráficas de usuario (GUI) que permiten a los usuarios interactuar con la computadora mediante ventanas, iconos y menús en lugar de comandos de bajo nivel.

Abstracción de dispositivos: El sistema operativo ofrece una abstracción de dispositivos, lo que significa que las aplicaciones pueden interactuar con dispositivos de E/S (como impresoras, teclados o mouse) a través de interfaces genéricas y no necesitan conocer los detalles técnicos de cada dispositivo.

Administración de archivos: El sistema operativo proporciona una forma de organizar y acceder a archivos y datos, lo que permite a los usuarios y aplicaciones interactuar con datos almacenados en dispositivos de almacenamiento sin preocuparse por la ubicación física de los archivos.

Gestión de procesos: El sistema operativo administra la ejecución de procesos y aplicaciones, lo que permite a múltiples aplicaciones ejecutarse de manera concurrente sin que los usuarios tengan que preocuparse por la gestión de la CPU.

Seguridad y control de acceso: El sistema operativo implementa mecanismos de seguridad y control de acceso que permiten a los usuarios y aplicaciones proteger sus datos y recursos, así como garantizar un uso seguro del sistema.

En resumen, el sistema operativo actúa como una capa de abstracción entre el usuario y el hardware, simplificando la interacción y permitiendo que los usuarios utilicen la computadora de manera más fácil y eficiente sin la necesidad de lidiar con los detalles internos del hardware.

15. ¿Es lo mismo el kernel que el sistema operativo?

En resumen, el kernel es una parte esencial del sistema operativo, pero no constituye la totalidad del sistema. El sistema operativo es un conjunto más completo de software que incluye el kernel y otros componentes que permiten que la computadora funcione de manera efectiva y que los usuarios interactúen con ella de manera conveniente. El kernel se encarga de las tareas más fundamentales, mientras que el sistema operativo en su conjunto ofrece una experiencia de usuario completa.

16. La memoria principal es un recurso del tipo multiplexada en el espacio

la memoria principal (RAM, Random Access Memory) no es un recurso multiplexado en el espacio. Más bien, se trata de un recurso de acceso aleatorio en el que los datos y programas se almacenan para su acceso rápido por parte de la CPU.

La memoria principal es una parte fundamental de un sistema informático y se utiliza para almacenar temporalmente datos y programas que están siendo utilizados por la CPU. Es un recurso crítico para el rendimiento de la computadora, ya que cuanto más RAM esté disponible, mayor será la cantidad de programas y datos que se pueden cargar y ejecutar simultáneamente.

La idea de "multiplexación en el espacio" se refiere a la técnica de compartir recursos entre varios usuarios o procesos asignando segmentos de tiempo o espacio a cada uno de ellos. Esto se usa comúnmente en sistemas compartidos, como la multiplexación en el tiempo en sistemas de comunicación, donde varios canales comparten el mismo medio, pero en diferentes momentos.

En contraste, la memoria principal se comparte entre los procesos de manera concurrente, pero no se multiplexa en el espacio de la misma manera que se haría con recursos compartidos de acceso múltiple en sistemas de comunicación. Cada proceso tiene acceso a su propia área de memoria en la RAM, lo que permite el acceso simultáneo y directo a los datos y programas que requieren.

17. El procesador en un sistema monoprocesador es un recurso del tipo multiplexado en el tiempo a cada proceso.

Sí, en un sistema monoprocesador, el procesador es un recurso que se multiplexa en el tiempo. Esto significa que se comparte entre varios procesos asignándoles segmentos de tiempo para su ejecución secuencial. La multiplexación en el tiempo permite que varios procesos compartan la única CPU física de manera efectiva, lo que da la ilusión de multitarea, aunque en realidad solo un proceso se ejecuta en un momento dado. La CPU se conmuta rápidamente entre procesos, lo que brinda a los usuarios la impresión de que múltiples procesos se ejecutan simultáneamente.

En resumen, en un sistema monoprocesador, el procesador se multiplexa en el tiempo para permitir la ejecución secuencial de múltiples procesos, pero no se ejecutan simultáneamente en paralelo en el procesador debido a la limitación de tener un solo núcleo de CPU.

18. Date se implementa con una system call?

Lamento la confusión anterior y gracias por señalarlo. En sistemas Unix y Unix-like, la función date se implementa como una llamada al sistema o una interfaz para acceder a la fecha y hora del sistema. xd

19. Un proceso tiene un stack en modo usuario y un stack en modo supervisor. Como no se usan a la vez, ocupan la misma dirección de memoria. (V o F)

Falso. En la mayoría de las arquitecturas y sistemas operativos, un proceso tiene un stack en modo usuario y un stack en modo supervisor que ocupan diferentes direcciones de memoria y se utilizan para propósitos diferentes.

20. El estado del proceso está en la PCB. (V o F)

Verdadero (V). El estado del proceso se almacena en la PCB (Block de Control de Procesos o Process Control Block) en un sistema operativo. La PCB es una estructura de datos fundamental que se utiliza para mantener información sobre cada proceso en ejecución o en espera en el sistema.

La información almacenada en la PCB incluye, entre otros datos:

Estado del proceso: Esto incluye si el proceso está en ejecución, en espera, listo o en otro estado.

Contador de programa (program counter): Indica la dirección de la próxima instrucción que se ejecutará en el proceso.

Registros de CPU: Almacena los valores de los registros de la CPU del proceso.

Información de asignación de recursos: Esto incluye información sobre la memoria asignada, dispositivos de E/S en uso y otros recursos del sistema.

Prioridad del proceso: Si el sistema operativo utiliza un esquema de planificación basado en prioridades.

Identificación del proceso: Número de proceso, nombre del proceso, identificador del usuario, entre otros.

La PCB es esencial para el control y la gestión de procesos en un sistema operativo. Permite que el sistema operativo conmute entre procesos, mantenga el estado y los contextos de los procesos, realice la planificación de la CPU y asegure un funcionamiento eficiente y seguro del sistema.

21. Un proceso crea a otro mediante un system call. (V o F)

Verdadero (V). En sistemas operativos, un proceso puede crear a otro proceso mediante una llamada al sistema (system call) específica. La llamada al sistema que permite la creación de nuevos procesos se conoce comúnmente como "fork" en sistemas Unix y sistemas similares.

La llamada al sistema "fork" crea una copia exacta del proceso que la llama, incluyendo su código, datos y estado. El proceso original se denomina "proceso padre" y el nuevo proceso creado se llama "proceso hijo". El proceso hijo comienza su ejecución a partir del mismo punto en el programa en el que se encontraba el proceso padre en el momento de la llamada "fork". A partir de ese punto, el proceso padre y el proceso hijo pueden seguir caminos de ejecución independientes.

La creación de procesos mediante llamadas al sistema es fundamental para la multitarea y la capacidad de ejecutar múltiples tareas concurrentemente en un sistema operativo. Los procesos creados pueden ser utilizados para realizar diferentes tareas y pueden comunicarse entre sí según sea necesario.

En resumen, un proceso puede crear a otro proceso utilizando una llamada al sistema específica, lo que permite la generación de procesos adicionales y es un concepto central en la multitarea de un sistema operativo.

22. La cola de procesos está en el disco. (V o F)

Falso (F). La cola de procesos (también conocida como cola de procesos en espera o ready queue) no se encuentra en el disco, sino en la memoria principal (RAM) de un sistema informático. La cola de procesos es una estructura de datos que se utiliza para mantener un conjunto de procesos que están listos para ser ejecutados por la CPU.

En la cola de procesos, los procesos esperan su turno para ser programados por el planificador de la CPU y ejecutados en la CPU. Los procesos en esta cola están en un estado listo para la ejecución, lo que significa que están esperando a que la CPU les asigne tiempo de ejecución. Cuando un proceso obtiene tiempo de CPU, se retira de la cola de procesos y comienza a ejecutarse en la CPU.

El disco se utiliza principalmente para almacenar programas, datos y archivos, pero no para mantener la cola de procesos en espera. La memoria principal (RAM) es donde se almacena la información sobre los procesos en ejecución y en espera para un acceso más rápido por parte del sistema operativo.

En resumen, la cola de procesos se encuentra en la memoria principal (RAM) y se utiliza para gestionar la planificación de la CPU y la ejecución de procesos, mientras que el disco se utiliza para el almacenamiento de datos y programas.

23. Cuando un proceso se crea, está en disco. (V o F)

Falso (F). Cuando se crea un proceso en un sistema operativo, este proceso generalmente se inicia en la memoria principal (RAM) y no en el disco. La creación de un proceso implica asignar recursos, establecer su contexto de ejecución y, en la mayoría de los casos, cargar el código y los datos del programa en la memoria RAM para que esté listo para ejecutarse.

Cuando se crea un proceso, el sistema operativo realiza varias tareas, como la asignación de espacio de memoria, la inicialización de registros y estructuras de control, y la carga del programa ejecutable en la memoria. Una vez que se ha realizado esto, el proceso se encuentra en un estado listo para ser programado por la CPU y se agrega a la cola de procesos en espera (ready queue) para que pueda ser ejecutado.

El disco se utiliza para almacenar programas y datos a largo plazo, pero no es el lugar donde se inicia un proceso. Sin embargo, es posible que el código y los datos del programa se almacenen en el disco antes de cargarlos en la memoria RAM al crear el proceso, pero el proceso en sí mismo comienza su ejecución en la memoria RAM.

24. El proceso padre crea al hijo en su propio espacio de direcciones. (V o F)

Falso (F). En la mayoría de los sistemas operativos, cuando un proceso padre crea un proceso hijo, el proceso hijo generalmente se ejecuta en su propio espacio de direcciones y se le asigna un nuevo espacio de direcciones. Esto se hace por razones de aislamiento y seguridad.

El proceso hijo es una copia del proceso padre, pero esta copia tiene su propio espacio de direcciones virtual. El espacio de direcciones virtual es la región de memoria en la que un proceso almacena sus datos y código. La copia del proceso hijo incluye una copia del código y los datos del proceso padre, pero se carga en una ubicación separada en la memoria.

El proceso hijo puede, en su mayoría, heredar algunas propiedades y recursos del proceso padre, como el entorno de ejecución, los archivos abiertos y otros recursos compartidos. Sin embargo, la separación de los espacios de direcciones virtuales garantiza que los procesos padre e hijo no interfieran directamente entre sí en cuanto a la memoria en la que operan.

La creación de un nuevo espacio de direcciones para el proceso hijo es fundamental para garantizar el aislamiento y la independencia de los procesos en un sistema operativo multitarea, lo que evita conflictos y problemas de seguridad.

25. Las tablas de archivos correspondientes a los archivos abiertos que está usando un proceso, forman parte de su contexto. (V o F)

erdadero (V). Las tablas de archivos correspondientes a los archivos abiertos que está utilizando un proceso forman parte de su contexto. El contexto de un proceso incluye toda la información necesaria para mantener y restaurar el estado del proceso en cualquier momento. Esto incluye información sobre los archivos que el proceso ha abierto y está utilizando.

Las tablas de archivos, que contienen información sobre los archivos abiertos por el proceso, forman parte de este contexto. La información en estas tablas puede incluir detalles como el descriptor de archivo, la posición actual en el archivo, el modo de acceso, entre otros.

Cuando un proceso se suspende (por ejemplo, cuando se produce un cambio de contexto o se realiza una conmutación de procesos), es esencial que todo su contexto, incluidas las tablas de archivos, se guarde adecuadamente para que el proceso pueda reanudar su ejecución más tarde en el mismo estado en el que se detuvo.

En resumen, las tablas de archivos asociadas a los archivos abiertos son parte del contexto de un proceso y se mantienen para permitir la reanudación de su ejecución sin perder información importante sobre los archivos que está utilizando.

26. La PCB se crea a partir que el proceso se carga en memoria. (V o F)

Falso (F). La creación de la PCB (Process Control Block) no está necesariamente vinculada a que un proceso se cargue en memoria. La PCB se crea cuando se crea un proceso, antes de su carga en memoria y ejecución. La PCB es una estructura de datos esencial que se utiliza para mantener información sobre el proceso y administrar su ejecución en el sistema operativo.

27. Luego de la system call fork, el proceso padre y el proceso hijo comparten la PCB. (V o F)

Falso (F). Después de que se realiza una llamada al sistema "fork" en un sistema operativo, el proceso padre y el proceso hijo no comparten la misma PCB (Process Control Block). Cada proceso tiene su propia PCB independiente.

La llamada al sistema "fork" crea una copia exacta del proceso padre, incluyendo su PCB, pero el proceso hijo tiene su propia PCB que se inicializa de manera independiente con la información adecuada. A partir de ese momento, el proceso padre y el proceso hijo operan de forma independiente, y cualquier cambio en la PCB de uno no afecta a la del otro.

La copia de la PCB en el proceso hijo es necesaria para mantener su propio estado y contexto de ejecución, y para asegurar la independencia y el aislamiento entre procesos. Cada proceso tiene su propia PCB para administrar su ejecución y recursos de manera independiente.

En resumen, después de una llamada al sistema "fork", el proceso padre y el proceso hijo tienen PCB independientes, ya que cada uno de ellos es un proceso separado con su propio estado y contexto.

28. Si no fuera por la E/S, los procesos no necesitarían system calls. (V o F)

Falso (F). Aunque las llamadas al sistema (system calls) a menudo se asocian con operaciones de entrada/salida (E/S), los procesos aún necesitarían las system calls para realizar una variedad de otras tareas y funciones, incluso si no estuvieran involucradas operaciones de E/S.

Las system calls son funciones proporcionadas por el sistema operativo que permiten a los programas de usuario interactuar con el hardware y realizar diversas tareas esenciales. Algunas de las tareas que requieren system calls incluyen:

Administración de memoria: Las system calls se utilizan para asignar y liberar memoria, así como para proteger y gestionar el espacio de direcciones virtuales de los procesos.

Planificación y administración de procesos: Las system calls permiten la creación, terminación y administración de procesos, así como la sincronización y la comunicación entre procesos.

Acceso a archivos y sistemas de archivos: Las system calls se utilizan para abrir, leer, escribir y cerrar archivos, además de crear, eliminar y administrar directorios.

Comunicación entre procesos: Las system calls se utilizan para la comunicación interprocesos, como el uso de colas de mensajes, semáforos y memoria compartida.

Control de dispositivos y hardware: Las system calls permiten la interacción con dispositivos de hardware, como controladores de dispositivos, interrupciones y configuraciones de hardware.

Aunque las operaciones de entrada/salida son un caso importante en el que se utilizan las system calls, el sistema operativo proporciona una amplia variedad de llamadas al sistema para habilitar muchas otras funcionalidades y operaciones esenciales en un sistema informático. Por lo tanto, los procesos aún necesitarían system calls incluso si no hubiera operaciones de E/S.

29. En modo supervisor, es posible acceder al espacio de direcciones de cualquier proceso. (V o F)

Si podria, pero el acceso al espacio de direcciones de cualquier proceso desde el modo supervisor está restringido y controlado por el sistema operativo para garantizar el aislamiento y la seguridad entre los procesos en un sistema operativo multitarea.

30. El contexto de un proceso es lo mismo que su espacio de direcciones. (V o F)

Falso (F). El contexto de un proceso y su espacio de direcciones son conceptos diferentes en la administración de procesos de un sistema operativo.

Contexto de un proceso: El contexto de un proceso se refiere a la información necesaria para mantener y controlar la ejecución del proceso. Esto incluye la información almacenada en la PCB (Process Control Block) del proceso, como su estado, registros de CPU, identificador único, información de recursos, prioridad y más. El contexto de un proceso se utiliza para administrar y reanudar la ejecución del proceso cuando se produce un cambio de contexto, como una conmutación de procesos.

Espacio de direcciones de un proceso: El espacio de direcciones de un proceso es la región de memoria virtual en la que el proceso almacena su código, datos y otros recursos durante su ejecución. Cada proceso tiene su propio espacio de direcciones, que es independiente de los espacios de direcciones de otros procesos. El sistema operativo se encarga de administrar y asignar el espacio de direcciones de cada proceso para garantizar el aislamiento y la protección entre procesos.

En resumen, el contexto de un proceso se refiere a la información sobre el estado y la administración del proceso, mientras que el espacio de direcciones de un proceso se refiere a la región de memoria virtual en la que el proceso almacena su código y datos durante su ejecución. Son conceptos diferentes en la administración de procesos de un sistema operativo.

31. Para implementar **prioridad dinámic**a o aging por inanición, se tiene en cuenta:

a) cuanto tiempo de CPU usó el proceso recientemente;

b) cuanto tiempo de espera tiene acumulado

Se, Ambos factores son fundamentales para determinar la prioridad de un proceso en un sistema que utiliza esta técnica. El tiempo de CPU utilizado recientemente puede influir en la prioridad de un proceso en función de su actividad, mientras que el tiempo de espera acumulado es esencial para evitar la inanición de procesos con baja prioridad. La combinación de estos dos elementos asegura que los procesos tengan una oportunidad justa de ser programados y ejecutados en el sistema operativo.

32. Un cambio de modo involucra un cambio de contexto.

Sí, un cambio de modo (también conocido como cambio de privilegio) involucra un cambio de contexto en el contexto de la administración de procesos en un sistema operativo.

Un cambio de modo se produce cuando un procesador cambia de un nivel de privilegio a otro, generalmente desde el modo usuario al modo supervisor (también llamado modo kernel) o viceversa. El modo usuario es el nivel de privilegio más bajo, mientras que el modo supervisor es el nivel de privilegio más alto.

Cuando se realiza un cambio de modo, se debe cambiar el contexto del procesador para que se pueda ejecutar código en el nuevo modo. Esto implica guardar el estado actual del proceso en ejecución (en el modo antiguo) en su PCB (Process Control Block) y cargar el estado del nuevo modo (en el modo nuevo) para que el código del sistema operativo o del kernel se ejecute con los privilegios adecuados.

El cambio de contexto es una operación esencial en la administración de procesos y es necesaria para garantizar que el sistema operativo pueda realizar tareas críticas de administración del sistema y proteger la integridad y la seguridad del sistema en su conjunto.

En resumen, un cambio de modo implica un cambio de contexto, que es necesario para cambiar entre niveles de privilegio en un sistema operativo y permitir la ejecución de código en el nuevo modo.

33. Un cambio de contexto involucra un cambio de modo.

No necesariamente. Un cambio de contexto y un cambio de modo son dos conceptos diferentes en la administración de procesos de un sistema operativo.

En resumen, un cambio de contexto y un cambio de modo son conceptos diferentes, y no todos los cambios de contexto implican necesariamente un cambio de modo.

34. Es lo mismo cambio de contexto que cambio de proceso?

Siii xdd

No, los términos "cambio de contexto" y "cambio de proceso" no son lo mismo, aunque están relacionados en el contexto de la administración de procesos en sistemas operativos multitarea. Aquí está la diferencia clave:

Cambio de contexto: Se refiere específicamente a la operación de guardar el estado actual de un proceso en ejecución y cargar el estado de otro proceso que se ejecutará a continuación. Esta operación es necesaria para permitir la conmutación o cambio entre procesos en la CPU. Implica la transición de la CPU de un proceso a otro al cargar el estado del próximo proceso desde su PCB (Process Control Block). El cambio de contexto es una parte fundamental del cambio de proceso.

Cambio de proceso: Se refiere al acto de cambiar de un proceso en ejecución a otro proceso en un sistema operativo multitarea. Esto incluye tanto el cambio de contexto como la transición real de la ejecución de un proceso al siguiente. El cambio de proceso implica la selección de un nuevo proceso listo para ejecutarse y la ejecución de su código en la CPU. El cambio de proceso es un proceso más amplio que involucra la planificación y la gestión de la ejecución de múltiples procesos.

Entonces, mientras el cambio de contexto es una parte esencial del cambio de proceso, no son lo mismo. El cambio de proceso abarca todo el proceso de selección y transición de la ejecución entre procesos, mientras que el cambio de contexto es una operación específica dentro de ese proceso más amplio.

35. Es lo mismo cambio de contexto que cambio de modo?

no

En resumen, el cambio de contexto y el cambio de modo son dos operaciones diferentes, aunque pueden estar relacionadas en el proceso de cambio entre procesos en un sistema operativo multitarea. El cambio de contexto se refiere a la transición entre procesos en términos de su estado de ejecución, mientras que el cambio de modo se refiere a la transición entre niveles de privilegio en términos de quién puede acceder y controlar el hardware del sistema.

36. Un fork exitoso produce cambios en la PCB del padre pues se almacena .... del hijo.

"Un fork exitoso produce cambios en la PCB del padre, pues se almacena **información** del hijo."

Espacio de direcciones: El espacio de direcciones del proceso hijo se inicializa con una copia del espacio de direcciones del proceso padre en el momento en que se realiza el "fork". Esto incluye tanto el código del programa como los datos del proceso.

Registros de CPU: Los registros de la CPU del proceso hijo se inicializan con los valores de los registros de la CPU del proceso padre en el momento del "fork". Esto permite que el proceso hijo continúe la ejecución desde el punto exacto en el que se encontraba el proceso padre.

Identificadores y atributos: Se copian identificadores y atributos del proceso padre al proceso hijo. Esto puede incluir información como el identificador de proceso (PID), prioridad, propietario, etc.

Contadores y temporizadores: Los contadores y temporizadores del proceso hijo se inicializan con los valores del proceso padre en el momento del "fork".

Manejo de archivos y E/S: El proceso hijo hereda los descriptores de archivos y cualquier estado de E/S que el proceso padre tenía en el momento del "fork".

Variables y estructuras de datos locales: El proceso hijo hereda las variables y estructuras de datos locales del proceso padre, lo que significa que ambos procesos comparten la información almacenada en la memoria.

La copia de esta información permite que el proceso hijo comience su ejecución de manera independiente y continúe desde donde lo dejó el proceso padre. Sin embargo, es importante recordar que, a partir de ese punto, ambos procesos tienen su propia ejecución y espacio de direcciones separados, por lo que pueden evolucionar de manera independiente.

37. En el mecanismo de manejo de memoria con particiones, el espacio de direcciones de un proceso está delimitado por los registros ........ y ..............

n el mecanismo de manejo de memoria con particiones, el espacio de direcciones de un proceso está delimitado por los registros "base" y "límite" (también conocidos como "base address" y "limit register" en inglés).

Registro Base (Base Register): Este registro indica la dirección base o inicial del espacio de direcciones del proceso en memoria. Define la dirección a partir de la cual el proceso puede acceder a su memoria asignada.

Registro Límite (Limit Register): Este registro indica el límite o tamaño del espacio de direcciones del proceso. Define hasta qué dirección puede el proceso acceder en su espacio de direcciones. La dirección más alta a la que se le permite acceder se calcula sumando el valor del registro base al valor del registro límite.

Juntos, estos dos registros permiten al sistema operativo y al hardware controlar el acceso y la protección del espacio de direcciones de cada proceso en un sistema con manejo de memoria por particiones. Establecen una región de memoria contigua para el proceso y limitan su capacidad para acceder a direcciones fuera de ese rango, lo que proporciona aislamiento y protección entre procesos.

38. El fork devuelve dos valores: ... al proceso hijo y ...... al proceso padre.

En resumen, el valor de **retorno de fork es 0 en el proceso hijo** y **el PID del proceso hijo en el proceso padre**. Estos valores permiten a los procesos padre e hijo identificarse y coordinar su ejecución de manera efectiva.

39. Un acceso no autorizado a memoria es detectado por: a) El S.O. b) El Hardware c) No puede detectarse

Un acceso no autorizado a memoria puede ser detectado por:

**b) El Hardware**

El hardware del sistema es el responsable de la detección de accesos no autorizados a memoria. Los sistemas modernos utilizan mecanismos de protección de memoria, como el bit de protección de página, tablas de páginas y unidades de administración de memoria, para controlar y supervisar los accesos a la memoria. Cuando un programa o proceso intenta acceder a una región de memoria a la que no tiene permiso o intenta realizar una operación no válida en memoria, el hardware detecta esta violación de acceso y puede generar una excepción o interrupción (como una violación de segmento o página no válida). Luego, el sistema operativo puede manejar esta excepción y tomar medidas apropiadas, como finalizar el proceso infractor o generar un error.

En resumen, el hardware es el encargado de la detección de accesos no autorizados a memoria y genera interrupciones o excepciones para notificar al sistema operativo de tales violaciones.

40. Las Systems Calls se ejecutan en “Modo Privilegiado”. V o F

Verdadero (V).

Las llamadas al sistema (System Calls) se ejecutan en "Modo Privilegiado" o "Modo Kernel" del sistema operativo. En este modo, el código del sistema operativo se ejecuta con privilegios elevados, lo que le permite realizar tareas críticas de administración del sistema y acceder a recursos protegidos, como el hardware y la memoria del sistema. Las llamadas al sistema son una forma en que las aplicaciones y los procesos de usuario pueden solicitar servicios o recursos del sistema operativo, y se realizan una transición de modo de usuario a modo kernel para que el sistema operativo pueda realizar estas solicitudes de manera segura y controlada.

41. Ante un cambio de contexto, indique cuáles de estos elementos se guarda en la PCB: a)tabla de páginas; b)pila de usuario; c)tabla de archivos abiertos; d) estado del proceso

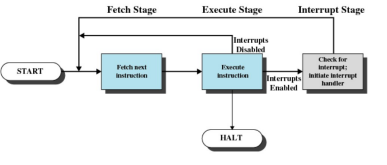
En un cambio de contexto, la PCB (Process Control Block) generalmente almacena:

b) Pila de usuario: La dirección y estado de la pila de usuario se almacenan en la PCB.

d) Estado del proceso: El estado del proceso, que incluye información como los valores de los registros de la CPU, el contador de programa y otros detalles relacionados con la ejecución del proceso, se almacena en la PCB.

La información sobre la tabla de páginas y la tabla de archivos abiertos suele manejarse de manera diferente. La tabla de páginas se gestiona como parte del manejo de la memoria virtual, y la información sobre archivos abiertos generalmente se mantiene en estructuras de datos separadas en lugar de la PCB. Por lo tanto, la tabla de páginas y la tabla de archivos abiertos no se almacenan directamente en la PCB en la mayoría de los sistemas operativos.

42. El chequeo de la existencia de una interrupción se realiza entre los pasos de “Fetch” y “Execute” de cada ciclo de instrucción



**No, el chequeo de la existencia de una interrupción generalmente se realiza después del paso "Execute"** de cada ciclo de instrucción. En el ciclo de instrucción típico de una CPU, hay varias etapas, que a menudo se dividen en "Fetch" (captura de la instrucción), "Decode" (decodificación de la instrucción), "Execute" (ejecución de la instrucción) y "Write Back" (escritura de resultados).

La comprobación de interrupciones se realiza generalmente en una etapa posterior a la ejecución de la instrucción, después de que se haya completado la ejecución de la instrucción actual. Esto permite que la CPU termine de ejecutar la instrucción actual antes de cambiar su flujo de ejecución para manejar una interrupción.

Cuando se produce una interrupción, la CPU generalmente completa la instrucción actual en curso, guarda su estado actual y luego salta a una rutina de manejo de interrupciones para tratar la interrupción. Esto garantiza que no se pierda información importante y que las interrupciones se manejen de manera ordenada y controlada. La comprobación de interrupciones y la transición a la rutina de manejo de interrupciones ocurren después de completar la ejecución de la instrucción actual.

43. El vector de interrupciones siempre debe estar en memoria

En resumen, el vector de interrupciones siempre debe estar disponible en memoria, pero esa memoria puede ser RAM, ROM, memoria flash u otros medios de almacenamiento, según las necesidades del sistema y la implementación específica. La clave es que sea accesible para el procesador cuando se produzca una interrupción.

44. Un system call fork, provocará cambio de contexto

una llamada al sistema "fork" suele provocar un cambio de contexto y un cambio de modo, ya que implica la suspensión de un proceso existente (cambio de contexto) y la ejecución de código del sistema operativo con privilegios elevados (cambio de modo) para crear y configurar el proceso hijo.

45. Un proceso swappeado en estado listo (ready to run) no compite por CPU.

En resumen, un proceso swappeado en estado "listo" todavía compite por la CPU, pero antes de que pueda ejecutarse, debe realizarse una operación de intercambio para traer sus datos nuevamente a la RAM. Una vez que los datos están en la RAM, el proceso está listo para ser programado y competir por la CPU junto con otros procesos en estado "listo".

**Osea no, tiene que volver a ram antes**

46. El scheduler de short term se ejecuta con menos frecuencia que el de long term.

En resumen, el planificador a corto plazo se ejecuta con más frecuencia que el planificador a largo plazo, ya que toma decisiones constantes sobre la ejecución de procesos en tiempo real, mientras que el planificador a largo plazo toma decisiones menos frecuentes sobre la admisión de procesos al sistema.

47. El cambio de contexto lo hace el scheduler de long term.

El cambio de contexto no es responsabilidad del planificador a largo plazo (long-term scheduler o admission scheduler). En cambio, el cambio de contexto generalmente es realizado por el planificador **a corto plazo** (short-term scheduler o CPU scheduler).

48. Cuando a un proceso se le termina su quantum, pasa a estado de espera.

Cuando el proceso termina su quantum (franja de tiempo) sin haber necesitado ser interrumpirlo por un evento, pasa al estado de ready, para competir por CPU, pues no está esperando por ningún evento...

49. El scheduler de medium term es quien decide el cambio entre nuevo y ready.

Noo, El scheduler de LONG term es quien decide el cambio entre nuevo y ready.

50. El scheduler de short term es quien hace pasar al proceso de estado ready a running.

Si, El scheduler de corto plazo lo eligió para asignarle CPU Tendrá la CPU hasta que se termine el período de tiempo asignado (quantum o time slice), termine o hasta que necesite realizar alguna operación de E/S

51. En la planificación de CPU se trata de maximizar la productividad, minimizar el tiempo de respuesta.

Por lo tanto, la planificación de CPU busca **un equilibrio** entre estos objetivos para satisfacer las necesidades específicas del sistema y sus usuarios. El equilibrio entre la maximización de la productividad y la minimización del tiempo de respuesta puede variar según el tipo de sistema y sus requisitos.

52. El tiempo de retorno, es el tiempo desde que se inicia hasta que termina, sumando cpu, espera en colas.

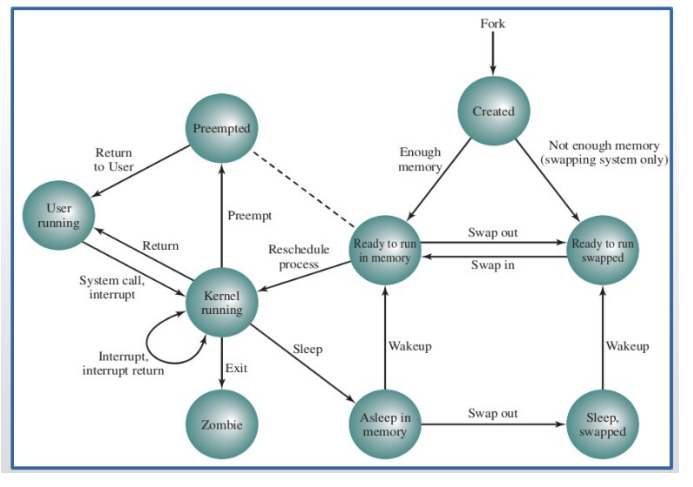
En resumen, el tiempo de retorno abarca todo el período desde que el proceso comienza su ejecución hasta que termina, y es una medida importante para evaluar el rendimiento de un proceso y la eficiencia del sistema operativo en la gestión de procesos.

53. Supongamos que un proceso está en espera swappeado y se cumple el evento por el que estaba esperando. El proceso queda en estado de listo en memoria secundaria.

Pues si, pero de ahí tiene que pasar a memoria principal para estar ready. Esta listo pero no realmente LISTO xd

54. Según el diagrama visto: puede un proceso pasar del estado de nuevo (creado) a listo swappeado? SI – NO

Si xd



55. Un proceso puede pasar de esperar en memoria secundaria a esperar en memoria principal.

No xd, pero si a la inversa. Sino deberia hacer todo un cliclo, espera secundaria, swap in, listo, running y espera en memoria.

56. El scheduler de medium term maneja el grado de multiprogramación.

no xd, ayuda(reduce). Pero de eso se encarga el long term

57. El disco permitió implementar la planificación de procesos.

Nada que ver papu

58. En un sistema monoprocesador, cuando se atiende una interrupción (se ejecuta una rutina de manejo de interrupciones) todos los procesos quedan en espera.

Ponele...

mientras el proceso que estaba en ejecución antes de la interrupción se suspende temporalmente, no todos los procesos quedan en espera simultáneamente. Otros procesos en estado de "listos" pueden competir por la CPU y ser seleccionados para su ejecución una vez que se complete la rutina de manejo de interrupciones. La idea es minimizar el tiempo de suspensión del procesador debido a interrupciones y permitir que otros procesos sigan ejecutándose en la medida de lo posible.

59. En un ambiente con procesos interactivos y batch, que maneja colas multinivel. ¿Conviene usar algoritmos apropiativos?

Si, porque los interactivos REQUIEREN algoritmos apropiativos, en los de batch pueden usarse no apropiativos.

60. Indique cuál es la combinación que representa la sucesión de actividades que realiza el dispatcher: a) Cambio de contexto; c) Salto a primer/próxima instrucción a ejecutar; b) Cambio de Modo; d) Carga en memoria del proceso elegido

A y C.

Cabio de modo lo hace SO, despues de interrupcion

Carga a memoria lo hace el Loader.

61. Indique que puede ocurrir cuando solamente se tienen muchos procesos orientados a I/O: a) Se incrementa el uso de CPU; b) Se saturan las colas de dispositivo;

La b xd

62. Cuando se carga un proceso en memoria, se hace en modo usuario.

No xd, se pasa a supervisor para que el loader lo cargue al que se selecciono

63. La dirección que se carga en el PC es una dirección física

No necesariamente, pero digamos que es virtual

64. En las particiones dinámicas siempre es mejor la opción worst fit para la asignación de particiones.

Pues por algun motivo si

No, ese se usa en particiones fijas.

65. ¿Quién resuelve una dirección en la que interviene el contenido del registro de reubicación y una dirección lógica?

Por lo tanto, en la práctica, la MMU es la que realiza la traducción de direcciones en tiempo real, mientras que el sistema operativo establece y mantiene las estructuras de asignación de memoria y configura la MMU para realizar estas traducciones de manera adecuada.

66. ¿Cual es la ventaja de la paginación pura con respecto a cargar todo el proceso en memoria de forma contigua? Analizar ventajas y desventajas

Paginación Pura:

Ventajas:

Uso eficiente de la memoria: En la paginación pura, los procesos se dividen en páginas pequeñas y se cargan en la memoria por demanda. Esto permite un uso más eficiente de la memoria, ya que se asigna memoria solo para las páginas que realmente se necesitan en un momento dado. No es necesario cargar todo el proceso en memoria de una sola vez.

Mayor flexibilidad: La paginación pura permite una mayor flexibilidad en la gestión de la memoria. Los procesos pueden crecer o reducirse en función de las páginas que se necesitan, lo que facilita la administración de la memoria en sistemas con múltiples procesos en ejecución.

Desventajas:

Fragmentación Interna: Puede haber fragmentación interna, ya que las páginas pueden no utilizarse completamente. Esto puede desperdiciar un poco de memoria debido a los espacios vacíos en las páginas.

Overhead de la tabla de páginas: La gestión de una tabla de páginas para mantener el mapeo entre direcciones virtuales y físicas agrega un pequeño overhead en términos de uso de memoria y tiempo de acceso.

Carga de Todo el Proceso en Memoria de Forma Contigua:

Ventajas:

Sin Fragmentación Interna: La carga de todo el proceso en memoria de forma contigua evita la fragmentación interna, ya que se utiliza toda la memoria asignada a un proceso de manera eficiente.

Acceso rápido: Al estar en una ubicación contigua, el acceso a datos y código es más rápido, ya que no se requiere una tabla de páginas para la traducción de direcciones.

Desventajas:

Ineficiente para procesos grandes: Puede ser ineficiente para procesos grandes que no caben completamente en la memoria, lo que puede limitar la multiprogramación y aumentar la necesidad de cargar y descargar procesos con frecuencia.

Dificultad de asignación de memoria: Puede ser difícil asignar memoria de manera eficiente a procesos que no requieren toda la memoria asignada de una vez. Esto puede resultar en una utilización ineficiente de la memoria.

En resumen, la paginación pura es más eficiente en términos de uso de memoria y flexibilidad, especialmente en sistemas con múltiples procesos en ejecución. Cargar todo el proceso de forma contigua puede ser más eficiente en términos de acceso rápido a la memoria, pero puede ser menos flexible y menos eficiente para procesos grandes. La elección entre estos enfoques depende de las necesidades específicas del sistema y los procesos que se ejecutan en él.

67. En paginación los procesos utilizan direcciones lógicas que son necesarias traducir a direcciones físicas.

Sí, la afirmación en la pregunta es verdadera. En la paginación, los procesos utilizan direcciones lógicas que necesitan ser traducidas a direcciones físicas en tiempo de ejecución, y esta traducción es una parte esencial del sistema de gestión de la memoria

68. Qué información es necesario guardar en el entrada de la tabla de páginas en la paginación pura?