¿Qué es un proceso?
Un proceso es una entidad abstracta que representa la ejecución de un programa, se trata de un conjunto de instrucciones que se crea cuando un usuario solicita ejecutar un programa, desde ese momento se convierte en una entidad dinámica capaz de cambiar su estado, interactuar con otros procesos, consumir recursos y seguir una trayectoria de ejecución particular.
La estructura fundamental que necesita un proceso para que el kernel de un sistema operativo pueda utilizarlo es el bloque de control de proceso (PCB).
Esta almacena el identificador del proceso, el identificador del proceso padre, su estado actual, prioridad, las direcciones de memoria a las cuales puede acceder, valores de registro de la CPU como el contador de programa, entre otros.
Después es necesario un espacio de direcciones de memoria donde el proceso pueda almacenar secciones de código, secciones de datos, stacks, etc.
El kernel también necesita de colas de planificación donde almacenar las PCB de los procesos para poder determinar que procesos pueden esperar, ejecutar, matar o swappear.

En un ambiente multiprogramado ¿Qué estructuras de datos y funcionalidades se deberían agregar al kernel para implementar una planificación de la CPU Round Robin? ¿Se debe contar con apoyo de hardware?

Para implementar una planificación Round Robin en un ambiente multiprogramado, el kernel de un sistema operativo debe poseer una cola de planificación para procesos “Listos”, donde se mantengan los procesos preparados para ejecutar en la CPU.
Además, debe implementar un algoritmo o planificador apropiativo, es decir, que pueda quitar procesos de la ejecución de la CPU sin que estos hayan terminado su tarea, asignando la CPU a otro proceso, según el orden de la cola.
El algoritmo o planificador “Round Robin” asigna a cada proceso un quantum de tiempo (un periodo/intervalo de tiempo fijo) para que utilice la CPU. Requiere apoyo de hardware mediante un temporizador o reloj del sistema, que genere interrupciones periódicas cuando se cumple el quantum. La expulsión de un proceso ocurre por dos motivos principales: al finalizar su tiempo asignado, cambia del estado “Ejecutándose” al estado “Listo” y vuelve a la cola de planificación para procesos “Listos” para que pueda competir por la CPU; o bien, cuando se genera una interrupción por entrada y salida (por ejemplo, leer un archivo), el proceso voluntariamente abandona la CPU, pasa del estado “Ejecutándose” al estado “En espera” y, una vez que se cumpla la operación, vuelve al estado “Listo”, para competir en nuevamente por la CPU en la cola de planificación.

¿Cómo funciona la memoria virtual con paginación por demanda? Tener en cuenta estructuras de datos necesarias: para que se las utiliza y quien las mantiene, fallos de página, su resolución y apoyo del hardware requerido

La memoria virtual con paginación por demanda es una técnica que permite al sistema operativo ejecutar procesos sin cargar completamente su espacio de direcciones en la memoria principal. Solo se cargan las páginas necesarias (una página es una sección del proceso) en el momento en que se requieren.
El objetivo es evitar que todo el proceso esté cargado en la memoria principal todo el tiempo, permitiendo mantener mayor cantidad de procesos ejecutándose simultáneamente o ejecutar procesos que su tamaño es mayor al que la memoria principal puede soportar. En

este esquema, cada proceso tiene un conjunto residente o working set, que representa las páginas de su espacio de direcciones que se encuentran actualmente cargadas en memoria y que utiliza con mayor frecuencia. Mantener un working set adecuado ayuda a minimizar los fallos de página.
Cada proceso dispone de una tabla de páginas, mantenida por el sistema operativo, donde se registran las correspondencias entre direcciones virtuales/logicas y marcos físicos de memoria (un marco es una sección de la memoria principal). Además, el sistema operativo mantiene una tabla de marcos libres, y el hardware incorpora una unidad de gestión de memoria (MMU) que utiliza una TLB (caché de traducciones) para acelerar el acceso a memoria.
Cuando una página de un proceso quiere acceder a una dirección virtual, la MMU consulta en la tabla de páginas el bit de validez asociado a esa página, si el bit de validez está en 0, significa que la página no está en la memoria principal y se necesita cargar desde el almacenamiento secundario, esto es lo que activa la demanda de la página, desencadenando un fallo de página.
El sistema operativo responde deteniendo temporalmente el proceso, eligiendo un marco libre o reemplazando uno existente según el algoritmo de remplazo de páginas que se utilice, cargando la página requerida desde el almacenamiento secundario. Una vez completada la transferencia, se actualiza la tabla de páginas, se marca el bit de validez en 1 y reanuda la ejecución del proceso.

¿Para qué sirven la técnica de frecuencia de fallos de página y técnica del working set? Expliquen como funcionan y compárelas, busque similitudes entre ellas. Indique como se pueden utilizar cada una para la detección de trashing (hiperpaginación)

El trashing es un problema que ocurre cuando el sistema gasta más tiempo atendiendo fallos de página (intercambiando páginas entre la memoria secundaria y la memoria principal) que utilizando la CPU para ejecutar procesos.
Para prevenir y solucionar este problema, el sistema operativo emplea dos técnicas: working set y frecuencia de fallos de página.
La técnica del working set actúa de manera proactiva. Define, para cada proceso, el conjunto de páginas que ha utilizado recientemente dentro de una ventana de tiempo determinada. Si la suma de los tamaños de los working sets de todos los procesos activos supera la cantidad de marcos disponibles en la memoria principal, el sistema está en riesgo de trashing, ya que la memoria es insuficiente para mantener los conjuntos residentes de todos los procesos. Para resolverlo, el sistema operativo reduce el grado de multiprogramación, liberando marcos para los procesos que permanecen activos.
La técnica de frecuencia de fallos de página actúa de manera reactiva. Supervisa la tasa de fallos de página de cada proceso: si esta frecuencia es consistentemente alta y supera un límite máximo, se interpreta que el proceso no tiene suficientes marcos asignados. La solución es aumentar su número de marcos o suspenderlo temporalmente si no hay más memoria disponible.
Ambas técnicas se basan en el principio de localidad, que establece que las direcciones de memoria referenciadas recientemente tienden a ser utilizadas nuevamente en un corto periodo de tiempo.
Ambas buscan evitar el trashing al intentar garantizar que cada proceso tenga en memoria la cantidad suficiente de marcos que necesita para su ejecución actual.

Ambas son técnicas que utiliza el sistema operativo para administrar de manera dinámica la cantidad de marcos asignados a un proceso.

Dado el llamado a la SysCall de lectura de un archivo f ya abierto, read(f, buff, count), enuncie el flujo que sigue para resolverla, considerando al menos la participación de: Subsistema de archivos (filesystem), buffer caché y driver de dispositivo

El llamado a la SysCall read(f, buff, count) para leer un archivo ya abierto sigue el siguiente flujo:

1. El proceso en modo usuario ejecuta la llamada a la función read, lo que provoca un cambio de contexto al modo kernel.
2. El subsistema de archivos (filesystem) recibe la solicitud y traduce el requerimiento de alto nivel en operaciones físicas sobre el dispositivo de almacenamiento.
3. A partir del descriptor de archivo se accede a las estructuras del sistema de archivos (por ejemplo, el i-nodo) para determinar qué bloques de disco contienen los datos solicitados.
4. Antes de acceder al disco, el kernel consulta el buffer caché para verificar si los bloques requeridos ya se encuentran en memoria.
    - Si los datos están en el buffer cache, se copian directamente al buffer del proceso, evitando el acceso al disco.
    - Si los datos no están en caché, se genera una solicitud de lectura al driver del dispositivo.
5. El driver de dispositivo traduce la solicitud en comandos específicos de E/S que el hardware entiende y los envía al controlador del dispositivo.
6. Cuando la lectura del bloque termina, el controlador genera una interrupción, que es atendida por el manejador de interrupciones del kernel.
7. Los datos leídos se almacenan en el buffer caché y luego se copian al espacio de usuario en el buffer indicado (buff).
8. Finalmente, se realiza un cambio de contexto de regreso al modo usuario, y la llamada a read() retorna la cantidad de bytes leídos.

Defina todo lo necesario para realizar las transiciones vistas en la teoria (new, ready running y waiting)

Para que un proceso pueda cambiar de estado entre los vistos en teoria (new, ready, running y waiting) es necesario que el sistema operativo tenga distintas colas de planificacion, estas almacenaran las las estructuras de bloques de control de proceso (PCB), estas estructuras de se encargan de almacenar toda la informacion necesaria para la ejecucion de un proceso (identificador del proceso, identificador de su proceso padre, direcciones de memoria, su contexto, etc).

En cada estado estas colas de planificacion se catalogan en long, medium y short. Es decir, para almacenar todos los procesos recientemente creados se almacenan sus PCB en una cola de planificacion long, esto para poder almacenar la mayor cantidad de procesos creados, luego,

Durante el ciclo de vida de un proceso, el mismo puede atravesar distintos estados. Describa cada uno de ellos indicando que componentes (modulos) del Sistema Operativo intervienen y sobre que estructuras de datos se trabaja para implementar dicho modelo de estados.

Un proceso puede atravesar distintos estados durante su ciclo de vida:

- New: El proceso esta siendo creado. El sistema operativo reserva los recursos iniciales (espacio de memoria, identificacor del proceso, etc.) y crea su PCB (Bloque de control del proceso) un estructura de datos donde almacena todo lo necesarios para el proceso (estado, contexto, registros, información de memoria, etc) esta sera almacenada en la long term scheduler (una cola de planificación a largo plazo) donde el sistema operativo almacena todos las PCB de los procesos recien creados. Los modulos del sistema operativo que intervienen son el gestor de procesos que crea y administra las PCB y el gestor de memoria donde se reserva espacio.
- Ready: Cuando el proceso termina de instanciarse, se lo selecciona de la long term scheduler para entrar en la CPU, alli cambia de estado, esta preparado para poder competir por el uso de la CPU. La PCB del proceso es insertada en la short term scheduler (una cola de planificacion a corto plazo) donde espera su turno para usar la CPU.
- Running: Cuando la CPU este libre y el planificador elige al proceso de la short term scheduler, es insertado en la CPU para ejecutar su tarea. Se cambia de estado y el proceso comienza a ejecutar sus instrucciones. Este proceso saldra de la CPU si:
    - Termina su tarea donde su estado pasa a Terminated y muere.
    - Es expulsado por algun algoritmo de planificación apropiativa como Round Robin, donde la terea del proceso no termina de ejecutarse pero se lo extrae y se lo manda nuevamente a la short term scheduler, donde su estado pasa a Ready, para darle espacio a otro proceso y mantener la multiprogramación.
    - Sucede una interrupcion donde el proceso deba esperar a que se complete una operacion de entrada y salida para continuar su ejecución, en este caso se pasa al estado waiting y la PCB del proceso es insertada en la medium term scheduler (una cola de planificacion a corto plazo).
- Waiting: El proceso no puede continuar porque esta esperando a que se complete una operacion de entrada y salida, cuando termine la operación, el proceso saldra de la medium term scheduler y sera insertado nuevamente en la short term scheduler donde se cambiara el estado a Ready nuevamente
- Terminated: El proceso completo su ejecucion o fue abortado por el sistema operativo, se elimina su PCB y libera los recursos asignados.

Uno de los grandes problemas que puede tener la tecnica de paginacion es que el tamaño de la tabla de paginas crezca considerablemente. Indique cuales son las 3 formas que existen de organizar la tabla de paginas con el fin de minimizar el inconveniente mencionado.

Para que la tecnica de paginacion funcione, cada proceso necesita una tabla de páginas, donde se asocia cada pagina lógica del proceso con su marco fisico en memoria.

Estas tablas se almacenan en memoria principal, cuando crecen considerablemente, desperdiciamos memoria que podria utilizarse para la ejecución procesos. Las tres maneras de minimizar este inconveniente son:

Tabla de un nivel: Es lo que genera el problema de consumo de memoria, se trata de una estructura de datos que almacena una entrada por cada pagina logica posible. Es simple y ofrece un unico acceso a memoria por lo que es muy rapido, el problema es que consume demasiada memoria incluso si el proceso no utiliza todas las paginas.

Tablas multinivel: Se utilizan dos o más tablas, donde la tabla del primer nivel contiene punteros a tablas de segundo nivel, estas tienen la direccion logica de la tabla . Esta tecnica utiliza menos recursos ya que crea tablas de segundo nivel si son necesarias, en el caso de que el processo no utilice ciertas areas de memoria, la estructura correspondiente no es creada, ademas las tablas de segundo nivel pueden ser almacenadas en memoria secundaria, lo que ahorra todavia más espacio. El unico problema es que el acceso a una pagina requiere varias consultas sucesivas (una por nivel), lo cual aumenta el tiempo de traduccion.

Tabla invertida (Hashing): En lugar de tener una tabla por proceso, se utiliza una sola tabla para todo el sistema para toda la memoria fisica. Cada entrada representa un marco fisico, e indica que pagina logica y que proceso se encuentra almacenada allí. Para localizar una página, el sistema operativo utiliza una función de hash que transforma el número de la pagina y el identificador del proceso en un índice de la tablas, esto reduce el tamaño de la tablas pero incrementa el tiempo de busqueda y la complejidad de la gestion.

¿Qué es el sistema de archivos? Describa y defina sus objetivos.

El sistema de manejo de archivos (file system) es un componente esencial del sitema operativo encargado de gestionar la organizacion, almacenamiento, acceso y persistencia de la informacion en los dispositivos de almacenamiento secundario (discos duras, SSD, memorias USB, etc).

El sistema de archivos provee una abstraccion logica sobre el hardware, permitiendo al usuario y a las aplicaciones trabajar con archivos y directorios mediante operaciones de alto nivel como open, read, write o close, sin necesidad de conocer los detalles fisicos de como los datos se guardan, actualizan, recuperan o eliminan.

El sistema operativo delega las operaciones de bajo nivel a los controladores (drivers) de los dispositivos de almacenamiento.

Sus principales objetivos son:

- Organizar y estructurar los datos en archivos y directorios.
- Proveer persistencia, asegurando que la informacion se mantenga disponible incluso despues de apagar el equipo.
- Facilitar el acceso eficiente y seguro a los datos.
- Controlar los permisos y la proteccion de los archivos frente a accesos indebidos.
- Gestionar el espacio de almacenamiento, asignando y liberando bloques de manera optima.

El sistema de archivos actua como intermediario entre las aplicaciones y los dispositivos fisicos perminitendo una gestion ordenada, persistente y segura de la informacion.

¿Qué caracteristicas debe tener el hardware para que el sistema operativo pueda brindar a los procesos proteccion del uso de la CPU?

Para que el sistema operativo pueda brindar proteccion del uso de la CPU a los procesos, el hardware debe incorporar mecanismos que permitan controlar la ejecucion, limitar los privilegios de los programas y asegurar la multiprogramacion.

Modos de operación: el procesador debe permitir distinguir entre modo usuario y modo kernel. Para esto se utiliza el cambio de contexto entre modos. En el modo kernel, el kernel del sistema operativo se ejecuta en modo privilegiado y tiene acceso completo a todas las estructuras internas, como a la PCB de cualquier proceso y sus espacios de direcciones.

En el modo usuario, los procesos solo pueden acceder a su propio espacio de direcciones. El sistema operativo impone este limite como medida de seguridad y proteccion para evitar que un proceso interfiera con los espacios de direcciones y recursos de otro proceso.

Otros detalles que aseguran la proteccion del uso de la CPU son la utilizacion de mecanismos de interrupcion por reloj como el utilizado para el algoritmo de planificacion Round Robin, que permite medir el tiempo de ejecucion de cada proceso para mantener una multiprogramacion, es decir, que varios procesos tengan la oportunidad de tiempo de ejecucion en la CPU.

MULTIPLES CHOICE (por las deudas)