Sistemas Operativos I
Final 1/03/2011

1. Planificación de CPU:
   a. Explique qué es un planificador de CPU con desalojo. Ejemplificar.
   b. ¿Este tipo de planificadores posee alguna ventaja en sistemas interactivos? Explique
   c. ¿Este tipo de Planificadores posee alguna ventaja en sistemas de tiempo real? Explique.
   d. ¿Cómo influye un planificador de CPU con desalojo en la programación de procesos multithread respecto de un planificador sin desalojo?
2. En sistemas de archivos se considera overhead al espacio utilizado por las estructuras del sistema de archivos, es decir, todo lo que no son datos de archivos propiamente dichos:
   a. Compare el overhead en función del tamaño de sistema de archivos para FAT32 e i-nodos planteando un escenario con una partición chica y otro escenario con una muy grande para ejemplificar.
   b. Compare el overhead en función del tamaño de sistema de archivos para FAT32 e i-nodos planteando dos escenarios con archivos chicos y muy grandes para ejemplificar.
3. ¿Puede ocurrir fragmentación en la memoria RAM? En caso positivo explique qué tipos de fragmentación pueden ocurrir. ¿Qué tipos de sistemas de administración de memoria sufren de fragmentación? ¿Se puede eliminar la fragmentación? ¿Cómo?

RESOLUCIÓN:

1.

a. Un CPU scheduling con desalojo permite desalojar (expropiar) a un proceso de la CPU, para comenzar a ejecutar otro proceso de la cola de listos. Estos algoritmos de planificación son utilizados por los sistemas operativos de tiempo compartido (para asegurar la interacción continua entre el usuario y sus aplicaciones) y sistemas operativos de tiempo real suave (para asegurar que las tareas que requieran completarse con alguna restricción de tiempo no sufran inanición).
   Si fuera sin desalojo, una vez que la CPU fue asignada a un proceso, este la mantiene hasta que termina o pasa a estado de espera.

b. Ventajas en sistema interactivos:
   En sistemas interactivos, la CPU es asignada a los procesos por tiempos muy cortos. Una vez que finaliza este tiempo, se necesita sacar al proceso de la CPU para colocar a otro en su lugar. De esta manera el usuario puede interactuar con las aplicaciones teniendo la sensación que cada aplicación se ejecuta en su propio procesador, ya que no sufren retardos perceptibles por las personas.

c. Ventajas en sistema de tiempo real:
   Este algoritmo de planificación se utiliza en los sistemas de tiempo real suave, en donde no hay una restricción estricta de tiempo pero la demora del kernel necesita ser limitada. En estos casos se utiliza

la planificación con prioridad (algoritmo de scheduling con desalojo), y se le da a los procesos que deben ejecutarse en tiempo real la mayor prioridad, lo que permite que cada vez que el proceso pasa de la lista de espera a la de listos recupere la ejecución del CPU siendo desalojado el proceso actual.

d. Si el sistema operativo soporta los hilos a nivel de kernel, entonces los procesos multihilos se ven beneficiados con la planificación apropiativa (con desalojo) por los mismos fundamentos expuestos anteriormente: hilos interactivos e hilos que requieran ejecutarse en tiempo real.
Sin embargo, la apropiación del CPU tiene un problema fundamental: RACE CONDITION. Race Condition se llama al problema de inconsistencia de una o mas variables al ser accedidas concurrentemente por mas de un thread mediante operaciones no atómicas. Así, el programador de aplicaciones multihilos debe usar mecanismos de sincronización.

2.

a. El overhead en FAT esta dado por la tabla FAT y la región del directorio raíz. El directorio raíz contiene los metadatos de los archivos y directorios del sistema, por lo tanto su tamaño no se ve afectado por el tamaño de la partición, sino mas bien por la cantidad de directorios y archivos. La tabla FAT tiene un tamaño directamente proporcional a la cantidad de clusters de la partición, ya que mantiene una entrada por cada cluster, por lo que el overhead es considerable en grandes particiones.
El overhead en inodos esta dado por los “grupos de cilindros”. Cada cilindro, o grupos de cilindros consecutivos, tienen una porción consecutiva de bloques llamada “grupo de cilindro” que contiene la cabecera del cilindro, con información sobre el cilindro, como puntero a espacio libre, etc, y la tabla de inodos, que es una secuencia de bloques con los inodos que, preferentemente, tienen sus bloques de datos en el mismo cilindro de disco. El overhead es mayor cuanto más grande es la partición, por la cantidad de cilindros, pero no se compara con FAT que mantiene información por clusters y no por cilindros.
Resultado: es mejor usar inodos que fat para particiones grandes.

b. En FAT el overhead por archivo es el mismo para pequeños que para grandes archivos, ya que toda la información de acceso y manipulación de los archivos se encuentra en la tabla FAT de tamaño fijo. Por cada archivo se agrega un pequeño overhead en la región del directorio raíz por los metadatos asociados al archivo, pero esta es la misma para pequeños y grandes archivos.
En inodos, el overhead por archivo es mayor en grandes archivos, ya que se requieren bloques de índices para el acceso a los bloques de datos. Cuanto mayor es el archivo, mas bloques se destinan a índices.
Resultado: es más chico el overhead asociado a FAT que a inodos, respecto al tamaño de los archivos.

3. En la tarea de asignar espacio de memoria a los procesos, puede generarse fragmentación interna y externa en la memoria RAM. La fragmentación externa se da cuando existe espacio libre suficiente para cubrir una solicitud pero esta se encuentra fragmentada en un gran número de huecos pequeños. La fragmentación interna es como la externa pero se da dentro del espacio de memoria de los procesos.

Todos los sistemas de administración de memoria sufren de uno u otro tipo de fragmentación:

- Asignación contigua de espacio: sufre tanto fragmentación interna como externa.
- Paginación: sufre solo fragmentación interna.
- Segmentación: sufre fragmentación externa y algo de interna.
- Paginación con segmentación: sufre algo de fragmentación interna.

La fragmentación se puede eliminar mediante: compactación y asignación no contigua de espacio.

La compactación es un mecanismo tedioso que reorganiza los contenidos de la memoria para colocar junta toda la memoria libre en un bloque solo. Este esquema solo se puede aplicar si los procesos son relocalizables dinámicamente.

La asignación no contigua de espacio a las solicitudes de memoria es el mecanismo que utiliza la paginación y, por lo tanto, la memoria virtual. Consiste básicamente en mapear direcciones lógicas de los procesos en direcciones físicas a bloques no contiguos de memoria. De esta manera, los procesos ven el acceso a memoria como el acceso sin limitaciones a un espacio contiguo de memoria.