Respuestas Orientadoras e incompletas (en algunos casos) a las preguntas de revisión. Favor tomarlas como orientadoras.

Revisión – Unidad V – Gestión de Memoria

1. ¿Qué es "Gestionar Memoria"?

Dividir la memoria física principal del sistema entre el SO y los procesos de usuario. Contar en memoria con procesos no ociosos: asegurar un flujo razonable de procesos listos para consumir tiempo de cpu.

2. ¿Por qué el SO debe poder reubicar un proceso en memoria?

Porque no se sabe de antemano en qué dirección de memoria va a estar un proceso al momento de su ejecución.

3. Responda V o F: La reubicación implica que debe hacerse una traducción de las referencias que están dentro del código del programa a direcciones de memoria físicas.

Verdadero.

4. Responda V o F: La reubicación facilita la implementación de la protección de memoria.

Falso. La reubicación dificulta, incrementa la complejidad de cualquier mecanismo de protección de memoria.

5. Responda V o F: El SO es quien detecta y aborta un proceso que referencia -sin permiso- posiciones de memoria fuera de su espacio de direcciones.

Falso. Es el hardware quien detecta un fallo en la protección de memoria de un proceso que será abortado por la CPU.

6. Describa la técnica de overlays.

La tecnica consiste en mantener solo una porción del programa en memoria principal y luego ir cargando y descargando los restantes modulos de la aplicación bajo demanda. Por ejemplo, una aplicación podría implementarse como un modulo principal y n modulos adicionales. El modulo principal podría estar formado por un conjunto de funciones o librería a ser utilizada en todo el sistema, más el menú principal del sistema, más datos o recursos globales a todo el sistema. Luego cada módulo adicional se correspondería con cada uno de los modulos aplicativos (clientes, proveedores, compras, ventas, etc.). Algunos linkeditores daban soporte para la linkedición de aplicaciones a través de esta técnica (ejemplo: Blinker).

7. En los tiempos del SO DOS era muy común que un programador debiera usar una técnica de overlays para lograr la ejecución de programas "grandes", ¿Por qué cree Ud. que esto era requerido?

Existen varias razones: 1. El SO DOS no tenia soporte para Memoria Virtual, por lo tanto, no podía ejecutar procesos que tuvieran un tamaño mayor que la memoria principal. 2. DOS no usaba ni paginación ni segmentación ni una combinación de ambos; reservaba un area de memoria para el sistema operativo y el resto de la memoria quedaba libre para el usuario, en donde podía cargarse el programa a ejecutar (puesto que no tenia soporte de multriprogramación, se ejecutaba un programa por vez), hacía una asignación fija de memoria, la memoria máxima que manejaba el sistema era de 640 KB (aunque el computador tuviese mucha más memoria), a ello habia que restarle lo usado por el kernel del SO; el límite estaba dado por direccionamiento, el sistema no podía representar direcciones de memoria por encima de los 640 KB, limitado por el espacio reservado a las direcciones de memoria. Es de notar, que las CPU's sobre las cuales se implementó y se usó habitualemente este sistema operativo, ya contaban con soporte para memoria virtual. Hubo software de terceros que dieron soporte a memoria extendida para intentar superar esta limitación del sistema.

8. ¿Qué entiende por "fragmentación interna"?

Se refiere al espacio desperdiciado en el interior de un bloque. Por ejemplo, suponga que dividimos

a la memoria en n bloques de longitud m y ubicamos allí al proceso p, este proceso podrá ocupar 3 bloques de longitud m más otro bloque adicional con un resto de x bytes, en donde x <= m; la fragmentación interna sería igual a m - x.

9. ¿Qué entiende por "fragmentación externa"?

Continuando con el ejemplo del punto anterior, supongamos que ubicamos un proceso p que ocupa 3 bloques, un proceso q de 2 bloques y un proceso w de 3 bloques. Si luego termina la ejecución del proceso q, quedarán 2 bloques libres en la memoria (entre medio de p y w), si luego se pretende ubicar en memoria a un nuevo proceso z que ocupa 3 bloques, si la ubicación debe hacerse en forma contigua, implica que no se pueden reutilizarse el "hueco" de 2 bloques que nos dejo el proceso q, a este hueco lo llamamos fragmentación externa.

10. Responda V o F: El particionamiento fijo tiene como ventaja que los procesos pequeños usen eficientemente el espacio ocupado en memoria.

Falso. Los procesos pequeños tendrán una alta fragmentación interna.

11. Tanto en el particionamiento fijo como en el particionamiento dinámico se habla de particiones de igual y de distinto tamaño, entonces, ¿Cuál es la diferencia entre ambos?

La diferencia es que, en el particionamiento fijo se reserva espacio para el SO y el resto de la memoria se divide en porciones fijas; mientras que en el particionamiento dinámico la memoria se va asignado a demanda de los procesos en bloques de cantidad y tamaño variable.

12. ¿Cuál es el objetivo de un "algoritmo de ubicación" (placement algorithm)?

Decidir que bloque de memoria asignar.

13. ¿Qué diferencia hay entre "marco" (frame) y "página" (page)?

Marco se refiere a las porciones (pequeñas) en que se divide la memoria física principal.
Pagina se refiere a las porciones (pequeñas) en que se divide a los procesos.
Las páginas se asignan a marcos disponibles. Un marco contiene una pagina.

14. La reubicación implica lidiar con distintos tipos de direcciones, ¿Qué tipos de direcciones de memoria Ud. conoce?

Direccion Física = dirección de memoria física dentro de la memoria principal
Dirección Base o Registro Base = dirección de memoria física inicial de un proceso
Dirección Final o Registro Valla = dirección de memoria física final de un proceso
Dirección Relativa = desplazamiento a partir de otra dirección base
Dirección Absoluta = Dirección Base o Registro Base + Dirección Relativa
Dirección Lógica = es igual que la dirección física, referencia a una posición determinada de la memoria, pero en este caso, es independiente de la actual asignación puntual en memoria.

15. Complete la Frase:

"La traducción de direcciones de memoria se hace en tiempo de ejecución de aquellos procesos que se encuentran en estado ejecutando (running), en el registro base de la CPU se carga la dirección inicial del proceso en memoria, en el registro "valla"(bound register) de la CPU se carga la dirección final del proceso en memoria. Para traducir una dirección relativa en una dirección absoluta, se toma el contenido del registro base y se suma la dirección relativa; el resultado de la suma se compara con el registro "valla" (bound register); si está dentro de los límites del proceso entonces se ejecuta la instrucción y sino la CPU genera una interrupción, asegurando de esta forma la protección de memoria."

Con alguno de los siguientes términos (que pueden usarse 0, 1 o n veces):
dirección intermedia del proceso en memoria - bloqueado (blocked) - tiempo de compilación - listo (ready) - - - - - suspendido (suspend) - - registro intermedio - dirección absoluta - dirección física - dirección intermedia - - - el SO -

13. Responda V o F: La paginación permite cargar un proceso aunque no existan marcos libres

en forma contigua.

Verdadero.

14. ¿Cómo es posible que un proceso pueda ejecutarse aunque no se encuentre en forma contigua en la memoria?

Porque el SO mantiene una tabla de paginas por proceso y las referencias a memoria son traducidas por la CPU de direcciones lógicas (número de página + desplazamiento) a direcciones físicas (número de marco + desplazamiento).

15. Indique la opción correcta. El SO mantiene una tabla de páginas por proceso, en dicha tabla: a) se accede por número de marco (frame) y se obtiene la página; b) se accede por número de página y se obtiene el número de marco.

a) Incorrecto b) Correcto.

16. ¿Por qué razón se dice que usando paginación hay un mínimo de fragmentación interna a pesar de que la paginación utiliza páginas y marcos de longitud fija?

Porque los marcos y páginas son muy pequeños.

17. Defina, aclare, diferencie los siguientes conceptos: Particionamiento, Paginación, Segmentación.

Particionamiento es una técnica antigua (previo a la memoria virtual), no usada actualmente, para dividir a la memoria física principal en bloques de longitud fija o variable (cabe aclarar que estos bloques tenían un tamaño muy superior a una pagina o marco).

Paginacion es una forma de particionamiento tanto de la memoria física principal como de los procesos en pequeñas porciones de longitud fija llamadas paginas (en cuanto a los procesos) y marcos (en cuanto a la memoria).

Segmentacion es similar a la paginación, pero en este caso los programas y sus datos se dividen en n segmentos que pueden tener distinta longitud, esto es más acorde a como se estructuran los programas más que la memoria física principal. Los modulos de codigo que conforman los programas son más bien "segmentables" que "paginables", puesto que cada programa se conforma de n modulos que pueden tener distinta longitud.

18. ¿Qué entiende por "thrasing"?

Trashing o trasiego se da cuanto el sistema pierde mas tiempo haciendo swapping que ejecutando instrucciones. Esto ocurre cuando hay muchos procesos en ejecucion y los fallos de páginas son muy frecuentes (en estos casos, se dice que esta fallando el control de carga).

19. ¿Por qué usar memoria virtual?

Porque permite independizar al programador de los límites de memoria.

Porque permite la ejecucion de programas con un tamaño mayor que la memoria principal.

Porque los programadores ya no tendrán que hacer overlays.

Porque permite ahorrar memoria cargando solo unos pocas porciones del proceso que seran usadas en un corto plazo (acorde con el principio de proximidad).

...

20. ¿Por qué razón debo traducir las referencias a memoria en tiempo de ejecución (runtime)?¿Por qué no hacerlo antes y evitar esa sobrecarga?

Porque no es posible saber de antemano en dónde quedará ubicado el proceso. Un proceso puede comenzar su ejecucion en la direccion 100 luego bloquearse, swappearse a disco y luego volver a cargarse para continuar su ejecucion a partir de la dirección 2000, hasta que no esté nuevamente ubicado en memoria y listo para continuar su ejecución no se sabe exactamente cuál será su ubicación en memoria.

21. ¿Por que razón cree Ud. que se requiere soporte de hardware para la traducción de referencias? ¿Por qué no se encarga de eso el SO?

Porque si lo hiciera el SO sería demasiado lenta la ejecución de procesos.

22. ¿Que entiende por "conjunto residente" (resident set)?

Es el conjunto de paginas que actualmente tiene un proceso en memoria. ...

23. ¿Por qué razón (vinculada con el manejo de memoria) un proceso puede quedar en estado bloqueado?

Un proceso en ejecucion puede hacer referencia a una página que no esta dentro de su conjunto residente, ello provocará un fallo de pagina, entonces el SO puede bloquear el proceso, cargar de la memoria secundaria la pagina requerida y luego volver a poner en ejecución al proceso que había dejado bloqueado.

24. ¿Qué relación hay entre "el principio de proximidad/vecindad" (locality principle) y el concepto de "thrasing"?

A medida que el conjunto residente de un proceso se hace más pequeño, acorde con el principio de proximidad, la posibilidad de fallo de página se incrementa y en la medida de que tengamos muchos procesos en memoria con conjuntos residentes demasiado pequeños, el sistema puede caer en trashing debido a la gran cantidad de fallos de pagina.

25. Responda V o F: Una implementación eficiente de memoria virtual no requiere de soporte de hardware para paginación y segmentación.

Falso. Memoria Virtual requiere de soporte de HW para paginacion y/o segmentación.

26. Responda V o F: En un SO con soporte para memoria virtual, parte de las tablas de páginas de un proceso, así como también parte del código del proceso en ejecución residen en memoria virtual.

Verdadero.

27. ¿Por qué razón Ud. cree que se divide la tabla de páginas en dos niveles jerárquicos? ¿Por qué complejizar el código y requerir de dos accesos a la tabla de páginas para traducir una dirección?

Para evitar tener que lidiar con un solo nivel jerárquico que implique un número muy grande de páginas (crecen proporcionalmente con el espacio de direcciones de memoria virtual) que difícilmente pueda mantenerse por completo en memoria principal, buscar una única entrada allí puede implicar varios accesos. Utilizando una jerarquía de varios niveles facilita el mantener en memoria ciertas porciones de la tabla de página en memoria, la busqueda en cada nivel es rápida (puesto que implica un número menor de entradas) y desde allí se apunta al siguiente nivel, dirigiendo la búsqueda hacia el valor buscado. Esta situación es análoga al planteo de implementar un índice exhaustivo (una entrada por cada valor de clave) o un índice no exhaustivo (una entrada por vencidad de clave o grupo de valores de clave).

...

28. ¿Qué relación hay entre el "control de carga" y los conceptos de "swapping" y "thrashing"?

El control de carga determina el número de procesos residentes en memoria principal (nivel de multiprogramación); muy pocos procesos en memoria puede ocasionar que -en un momento dado- todos los procesos estén bloqueados y se gaste mucho tiempo en swapping; demasiados procesos en memoria implica menos memoria principal para proceso, lo cual, puede llevar a un tamaño inapropiado del conjunto residente, provocando fallos de página muy frecuentes y ello llevará al trashing o trasiego (se pierde más tiempo haciendo swapping que ejecutando instrucciones).

29. ¿Por qué razón se utilizan las "tablas de páginas invertidas" (Inverted Table Pages)?

Debido a que el sistema tiene que lidiar con un número de páginas muy grande, se puede acelerar la búsqueda utilizando una ténica de hashing (al igual que ocurre con los archivos de acceso directo, en donde, en promedio, ubicar el registro destino esta en el orden de magnitud O(1.5)); se aplica una función de hashing que recibe como argumento el número de página a buscar y ésta devuelve la dirección en la tabla invertida, se accede a la tabla invertida, si es la pagina buscada se obtiene el marco (frame destino); si no es la página buscada puede tratarse de una página sinónima y requiere del puntero chain para acceder a otra entrada de la tabla invertida y ver si se trata ahora de la página buscada; así hasta encontrar el marco o bien generar un fallo para cargar otra porcion de la tabla

invertida a memoria principal.

30. ¿Para que sirve la función de dispersión (hash function)? ¿Cómo funciona? ¿Por qué existe un atributo llamado "chain" dentro de la tabla de páginas invertidas?

Idem anterior. Debido a como funciona un algoritmo de hashing. Idem a lo que sucede con archivos de acceso directo. La funcion de dispersión transforma clave a buscar (en este caso número de página) en un desplazamiento dentro del archivo de datos (en este caso, desplazamiento dentro de la tabla invertida). La funcion de dispersion puede implementarse de varias formas, un algoritmo posible es el del cubo, en donde la clave a buscar se divide por el tamaño del archivo y se obtiene el resto de la división como desplazamiento. El problema es que todos los algoritmos de dispersion pueden provocar claves sinónimas (2 o más paginas distintas que da como resultado el mismo desplazamiento), por ello se requiere de algún algoritmo para el manejo de claves sinónimas, una posibilidad es hacer una lista enlazada de registros con claves sinónimas, para ello se necesita del puntero chain.

...

31. ¿Cuál es el objetivo del "buffer de traducción anticipada" (Translation Lookaside Buffer (TLB))? ¿En dónde se encuentra? ¿Qué relación tiene con " el principio de proximidad/vecindad" (locality principle)?

Toda referencia a memoria virtual implica dos accesos físicos a memoria principal (1. en la tabla de páginas 2. armar la dirección física y accederla en memoria principal) el buffer de traducción anticipada (TLB) tiene como objetivo acelerar este proceso, poniendo las entradas de la tabla de páginas más recientemente usadas en una memoria de alta velocidad (cache), acorde con el principio de proximidad, se supone que éstas páginas tienen una mayor probabilidad de ser usadas en el corto plazo. De esta forma, primero se busca en el TLB y luego (si hay fallo de TLB) se busca en la tabla de páginas.

32. ¿Qué complicaciones hay vinculadas con el tamaño de la página?

Implica un dilema entre fragmentación interna Vs cantidad de entradas en la tabla de páginas. A menor tamaño de la página, menor fragmentación interna, pero ello también implica mayor cantidad de páginas requeridas.

33. Para discutir: Cuando se combinan Segmentación y Paginación, la secuencia de traducción es: Dirección Virtual - Segmentación - Paginación - Dirección Física ¿Por qué cree Ud. que esto es así? ¿Por qué no hacer: Dirección Virtual - Paginación - Segmentación - Dirección Física?

Se considera que ello es así por una cuestión de facilidad de implementación, puesto que los procesos están estructurados en términos de modulos de software y éstos son más bien "segmentables" que "paginables". Los módulos no son de longitud fija (como las páginas) sino que un proceso puede estar formado por n modulos de distintos tamaños cada uno. Por otro lado, el llevar todo en términos de páginas, que son bloques de igual longitud, facilita el desarrollo de algoritmos más sofisticados para manipularlas. Esto último sería más complicado si debiéramos trabajar con segmentos.

...

34. Responda V o F: Cualquier proceso puede accer al espacio de memoria del Dispatcher.

Falso. Fallo de protección de memoria.

35. Cuando se diseña un SO y se decide implementar soporte para memoria virtual, se deben implementar una serie de políticas relacionadas con: ¿cuándo traer una página a la memoria? ¿dónde ponerla? ¿cuando reemplazarla por otra?. ¿A qué políticas nos referimos?

¿cuándo traer una página a la memoria? --> Politica de recuperación
¿dónde ponerla? --> Politica de reubicación
¿cuando reemplazarla por otra? --> Politica de reemplazo

36. ¿Cuál es el principal objetivo de las políticas descriptas en el punto anterior?

Deben ser consideradas en el diseño de la gestión de memoria de todo SO, el objetivo principal es el rendimiento, minimizar la ocurrencia de fallos de página.

37. Si existe un "algoritmo de reemplazo optimo" (Optimal Replacement Algorithm) ¿Para qué lidiar con otros algoritmos? ¿Por qué no usar este y listo?

Debido a que el algoritmo óptimo es un algoritmo teórico (no hay implementaciones del mismo), sirve como medio de comparación para otros algoritmos.

38. ¿Qué diferencia hay entre "una estrategia de reemplazo de alcance local" (local scope replacement strategy) y una "estrategia de reemplazo de alcance global" (global scope replacement strategy)?

Ante una fallo de página, se puede optar por una estrategia de reemplazo de alcance local (selecciona páginas a reemplazar dentro de las páginas del proceso que genero el fallo) o bien de alcance global (se consideran todas las páginas en memoria principal que no estén bloqueadas, sin importar a qué proceso pertenecen).

39. ¿Cuándo debo aplicar una "política de suspensión" (suspension policy)?

Cuando debo reducir el grado de multiprogramación.

40. ¿Cuándo debo aplicar una "política de limpieza" (cleaning policy)?

Cuando debo decidir cuándo se debe escribir en disco una página que ha sido modificada.