# Repaso

## Segundo Parcial Teórico

## Memoria (2da parte), Archivos, Buffer Cache

### 1. La paginación por demanda es una forma de implementar memoria virtual.

Si. en la MV con paginacion:
- Cada proceso tiene su tabla de paginas.
- Cada entrada en la tabla referencia al frame en el que se encuentra la pagina en la memoria principal.
- Cada entrada en la tabla de paginas tiene bits de control, entre otros, el Bit V (indica si la pagina se encuentra en memoria o no) y el Bit M (indica si la pagina fue modificada, si es asi se debe reflejar los cambios en memoria secundaria).

La memoria virtual es agrandar la memoria principal con la memoria secundaria (área de swap).

### 2. En la administración segmentada de memoria hay una tabla general para todo el sistema.

No. Puede haber, hay dos alternativas, una única tabla para todos los procesos o una tabla por cada proceso.

### 3. Para solucionar un page fault, habrá que hacer en algún momento un context switch.

Verdadero. El proceso que produce el page fault queda en estado de espera y el scheduler de short term elige otro proceso para que sea ejecutado, allí es cuando se produce el cambio de contexto.

### 4. La resolución de direcciones en el momento de la carga facilita la administración paginada.

Falso. La única resolución que ayuda es la resolución al memento de ejecución.

### 5. Si un proceso contara con los frames que necesita, no tendría ningún page fault (fallo de página).

Verdadero. Si el SO dijera que hay tantos marcos libres como el proceso solicita se va a cargar todo. Porque también seria valido que aunque aya marcos libres no cargue todo el proceso sino que espere los page fault para cargar de a una página.

### 6. La tabla de páginas es parte de: el contexto / la PCB / el espacio de direcciones / la pila del proceso.

El contexto. La PCB no, en la PCB está la dirección de la tabla de paginas y en el espacio de direcciones tampoco, porque es todo lo referenciado en modo usuario por lo tanto no

podría estar la tabla de paginas allí. En la pila del proceso tampoco.

7. El tamaño de la tabla de segmentos depende del tamaño del proceso.

No, puedo tener un proceso muy grande y que los segmentos sean muy grandes, por lo que la tabla va a ser chica.

8. Si disminuyo la cantidad de bits del desplazamiento de una dirección de memoria, los frames serán más chicos.

Verdadero.

9. La cantidad máxima de páginas en memoria depende sólo del tamaño del proceso.

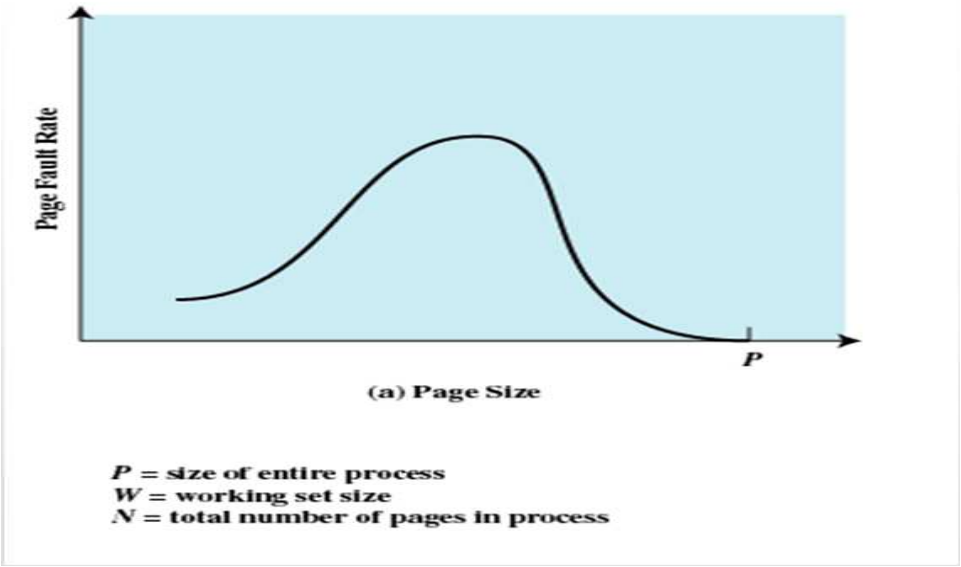
No, la cantidad máxima de paginas en la memoria depende del criterio de asignación de marcos de la memoria.

10. El bit de modificación relacionado con una página generará una I/O adicional.

Si, generá una E/S adicional.

11. Analice tamaños de página y page fault.

Cuando el tamaño de la pagina es muy grande la tasa de page fault va a tender a ser muy chica ya que el proceso va a tener pocas paginas, si las paginas son muy chicas la tasa de page fault no aumenta mucho ya que es muy probable que aya muchas paginas del proceso en memoria, cuando la tasa de page fault tiende a aumentar notablemente es cuando el tamaño de la pagina tiene un tamaño intermedio




12. Relación de tamaño de página, de proceso, de tabla de paginas según la arquitectura de la dirección.

13. La tabla invertida usa correspondencia directa.

No, la correspondencia directa es la utilizada por la tabla de paginas o segmentos. La asociativa (tabla invertida) es cuando se tiene una clave para buscar y acceder, es una sola tabla para todos los procesos.

14. Qué es el working set? Es siempre el mismo?

Working set: Conjunto de paginas de un proceso que en un momento dado está en memoria. La medida del working set es la cantidad de paginas que tiene el proceso en un momento dado.

No, el working set cambia, cuando se produce un page fault por ejemplo se produce un cambio en el working set, ya que se va a agregar una pagina a este,  si se swapea una pagina, pasa lo mismo.

15. Qué consecuencias puede tener la hiperpaginación.

Como consecuencia, hay una baja importante de performance en el sistema.

16. Que pasa si el delta elegido es muy chico? Y si es muy grande?

Ventana de working set (delta): Las referencias de memoria mas recientes.

Si tomo un delta muy grande voy a tener una noción mayor de que paginas son referenciadas, caso contrario en un delta muy chico.

17. Diferencia entre reemplazo global y local.

En el reemplazo global a la hora de elegir una pagina victima se puede elegir cualquier pagina que se encuentre en memoria, que pertenezca a cualquier proceso, en cambio en el reemplazo local la pagina victima que se elija tiene que ser del mismo proceso que produjo el page fault.

18. Con el reemplazo local no cambia la cantidad de frames asignados al proceso.

Verdadero. Ya que se va a sacar una pagina para poner otra correspodiente al mismo proceso.

19. Diferencia entre asignación equitativa y proporcional.

Reparto equitativo: Se asigna la misma cantidad de marcos a cada proceso.

Reparto proporcional: Se asignan marcos en base a la necesidad que tiene cada proceso.

20. Secuencia de resolución de page fault incluyendo TLB y cache común de memoria.

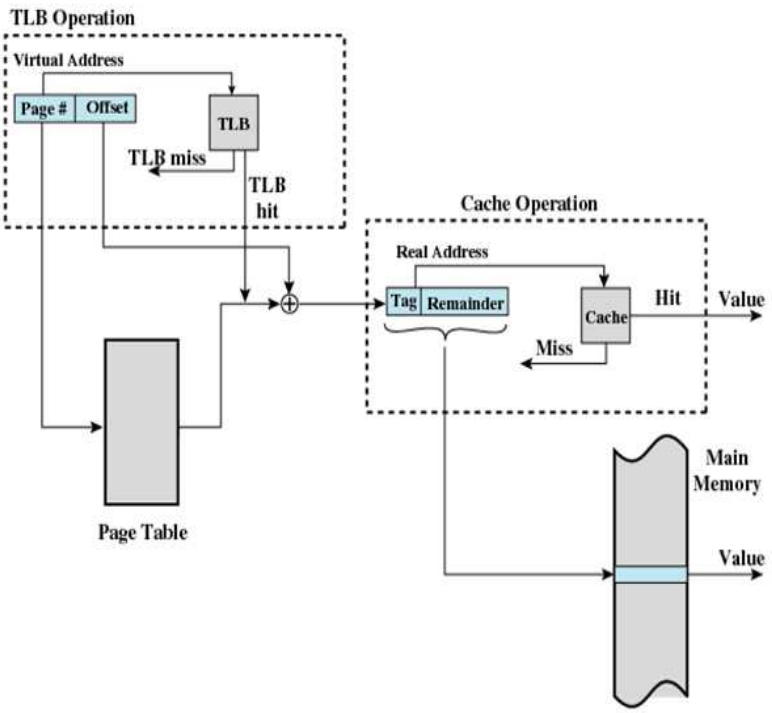



Figure 8.10 Translation Lookaside Buffer and Cache Operation

21. En cuanto estados del bit M y R: Cual seria la página ideal para elegir como pagina victima?

Cuando los dos están en 0.

Orden de preferencia:
R M
0 0
0 1
1 0
1 1

22. EL ancho de banda de un disco es muy importante en la performance de la paginación por demanda .

Si.

23. En qué momento se hace el chequeo sobre si el usuario puede acceder a un archivo: en el open? en cada read? en cada write?

En el open, es una system call, se chequea si el archivo existe y si tiene los permisos para ese usuario.

24. Un archivo de gran actividad, será de mucha volatilidad.

No, porque puede ser que tenga mucha referencia (actividad) y no volatilidad (modificación).

25. En Unix System V: puede modificarse el header del archivo sin modificar el archivo en sí?

Si, si se modifican los permisos por ejemplo, se modifica el header y no el archivo.

Si modifico el nombre, no accedo ni al header ni al archivo, sino al directorio.

26. En Unix System V: puede modificarse el archivo sin modificar su header?

No, porque se modifica la fecha en el header.

27. En Unix System V se verán beneficiados en performance los archivos cuyo contenido pueda ser referenciado por los 10 primeras direcciones de bloque que están en su inodo.

Verdadero. Están de esa manera (directo, indirecto simple, doble, triple) porque se trata de beneficiar a los archivos mas pequeños.

28. En un archivo como el del punto anterior, el acceso random puede realizarse accediendo directamente al bloque que necesito, sin leer los precedentes.

Verdadero, ejemplo: si cada bloque mide 1 KB referencia a un bloque de 1024 Bytes, si tengo que ir a la dirección 1100, no accedo al bloque 1, voy directamente al 2 (1100 / 1024).

29. Para poder ubicar en el disco un bloque perteneciente a un archivo: es necesario saber qué cantidad de inodos hay por bloque?

Si.

30. Puede asignarse un bloque a un archivo sin acceder previamente al superblock?

No, porque se le puede asignar un bloque que tenga asignado otro archivo.

31. Se puede acceder a un archivo sin acceder a su inodo.

32. Al crear un archivo en un filesystem, indique qué se modifica: directorio al que pertenece, superblock, lista de inodos de todos los filesystems.

Se modifica el directorio al que pertener y el superblock.

33. Puedo crear un archivo en un filesystem no montado?

No.

34. Todos los filesystems de un disco deben tener el mismo tamaño de bloque.

35. Cuando un archivo se borra, se ponen en cero los bloques

Falso, cuando un archivo se borra se ponen los bloques como disponibles.

36. La estructura del filesystem define el tamaño máximo del archivo.

Verdadero.

37. La estructura del filesystem define la longitud máxima del nombre.

Verdadero.

38. En la estructura de Buffer cache vista, un buffer puede estar ocupado y delayed write a la vez.

DW: mientras estaba en la memoria fue modificado.

Si, lo está usando un proceso y fue modificado antes.

Libre y DW también puede suceder, fue modificado antes.

39. El inodo de un archivo que se está usando debe estar en algun buffer del buffer cache.

Verdadero.

40. Un buffer delayed write puede volver a estar ocupado, si lo pide un proceso, pero antes debe grabarse a disco.

Falso, se graba por sincronización (osea cada tanto tiempo) o por ser el primero en la free list y algun proceso necesita cargar un bloque.

41. Para asignar un bloque a un archivo es necesario contar con el superblock en el buffer cache.

Verdadero, el superblock también es un bloque de disco.

42. Un proceso esperando por un buffer delayed write y ocupado, deberá esperar la escritura a disco antes de que se le asigne a él.

Falso, tiene que esperar que pase a libre.

43. Las hash queues sirven para buscar por un buffer en particular.

Verdadero.

44. La free list sirve para buscar por cualquier buffer.

Falso, la free list solo contiene los bloques libres.

45. Los buffers del buffer cache pueden tener distinto tamaño.

Verdadero.

46. No puede haber más de un proceso esperando por un buffer.

Falso.

47. Cuando un proceso libera un delayed write, es escrito a disco antes de ponerlo en la free list.

Falso.

48. Puede un buffer delayed write volver a estar ocupado?

Si.

49. Puede un buffer en la free list, poder estar ocupado?

No.

50. Si un buffer está primero en la free list y en ese momento lo pide un proceso: donde va cuando se libere?

Al final de la free list.

51. Si un proceso necesita un buffer y la free list está vacía, se aborta.

No, queda en espera.

52. Si una hash queue está vacía, se toma un buffer de otra cola.

Falso.

53. Para acceder a la tabla de páginas, ésta debe estar completa en el buffer cache.

Falso.

54. Conviene > cantidad de hash queues con pocos elementos, que < cantidad con +.

Conviene mayor cantidad de hash queues con pocos elementos para acceder mas rapido a los archivos.