RESUMEN TEÓRICO ISO

TEÓRICO VIEJO:

MEMORIA

1) En paginación, si disminuyo la cantidad de bits del desplazamiento de una dirección de memoria, los frames serán más chicos.

Verdadero. El tamaño del **desplazamiento** (offset) está determinado por la cantidad de bits que se le asignan, y este define el tamaño del **frame** (marco de página) porque:

- Si el desplazamiento usa **N bits**, el tamaño del frame es 2^N2N bytes.
- Si se **disminuyen** los bits del desplazamiento, el tamaño del frame también **disminuye**, ya que podrá direccionar un rango más pequeño dentro de cada página.

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

Falso. También depende de Tamaño de la memoria física, Tamaño de la tabla de páginas, Esquema de administración de memoria.

3) La tabla invertida proporciona acceso directo al marco buscado.

Falso. La tabla de páginas invertida no proporciona acceso directo al marco buscado, sino que utiliza una búsqueda para localizar la entrada correspondiente.

4) Qué consecuencias puede tener la hiperpaginación.

Disminuye el rendimiento, aumenta la latencia en la ejecución de procesos, uso excesivo del disco, menor capacidad de respuesta del disco, pérdidad de eficiencia en la multitarea.

5) En la técnica del Conjunto Trabajo ¿Qué pasa si el delta elegido es muy chico? ¿Y sí es muy grande?

Si es muy chico, no cubrirá la localidad. Si es muy grande, puede tomar varias localidades. Conclusión, lo ideal sería que el delta sea equilibrado.

5) Diferencia entre Reemplazo Global y Reemplazo Local

Reemplazo Global:

El fallo de página de un proceso puede reemplazar la página de cualquier proceso. El SO no controla la tasa de page-faults de cada proceso. Puede tomar frames de otro proceso aumentando la cantidad de frames asignados a él. Un proceso de alta prioridad podría tomar los frames de un proceso de menor prioridad.

Reemplazo Local:

El fallo de página de un proceso solo puede reemplazar sus propias páginas de su Conjunto Residente. No cambia la cantidad de frames asignados. El SO puede determinar cuál es la tasa de page-faults de cada proceso. Un proceso puede tener frames asignados que no usa, y no pueden ser usados por otros procesos.

6) Con el reemplazo local no cambia la cantidad de frames asignados al proceso

Verdadero.

7) Diferencia entre asignación equitativa y proporcional.

En la asignación **equitativa**, cada proceso recibe la misma cantidad de marcos, sin importar su tamaño.

✅ **Ventajas:**
✔️ Fácil de implementar.
✔️ Justo en términos de distribución uniforme.

❌ **Desventajas:**
❌ No considera las necesidades reales de los procesos.
❌ Puede desperdiciar memoria en procesos pequeños o causar fallos de página en procesos grandes.

En la asignación **proporcional**, Cada proceso recibe marcos en proporción a su tamaño.

✅ **Ventajas:**
✔️ Distribución más eficiente de la memoria.
✔️ Reduce el riesgo de fallos de página en procesos grandes.

❌ **Desventajas:**
❌ Puede perjudicar a procesos pequeños si se les asignan muy pocos marcos.
❌ Requiere cálculos adicionales en tiempo de ejecución.

8) Secuencia de resolución de page fault incluyendo TLB.

🔁 Resumen del Flujo

1️⃣ **Intento de acceso a memoria** →
2️⃣ **Búsqueda en TLB**

- ✅ **Si está en TLB** → Accede directamente.
- ❌ **Si no está en TLB** → Buscar en la tabla de páginas.
  3️⃣ **Búsqueda en tabla de páginas**
- ✅ **Si está en memoria** → Actualiza la TLB y accede.
- ❌ **Si no está en memoria** → Page Fault.
  4️⃣ **Carga de página desde disco**
  5️⃣ **Actualización de estructuras (tabla de páginas y TLB)**
  6️⃣ **Reintento de la instrucción**

9) En cuanto a los estados del bit M y R: ¿Cuál sería la página ideal para elegir como página victima?

Lo ideal sería ambos en 0. Pero sería aceptable que sea R=0 y M=1.

ARCHIVOS

10) 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?

El chequeo principal es en el open, después solamente se chequea si se cambian los permisos luego del open.

11) En Unix System V: ¿puede modificarse el i-nodo del archivo sin modificar el archivo en sí?

EL I-NODO SI SE PUEDE CAMBIAR SIN CAMBIAR EL CONTENIDO DEL ARCHIVO. AL REVÉS NO.

12) En Unix System V se verán beneficiados en performance los archivos cuyo contenido pueda ser referenciado por las 10 primeras direcciones de bloque que están en su i-nodo.

Verdadero.

✅ **Archivos pequeños que caben en los 10 punteros directos del i-nodo tienen mejor rendimiento** porque se accede a sus datos en un solo paso.
❌ **Archivos grandes, que usan punteros indirectos, sufren mayor latencia** porque requieren múltiples accesos a disco para localizar los datos.

12) En Unix System V, el acceso random a un archivo puede realizarse accediendo directamente al bloque que necesito, sin leer los precedentes.

Verdadero. En **UNIX System V**, los archivos están organizados en **bloques de datos**, y el **i-nodo** del archivo contiene punteros que permiten acceder directamente a estos bloques.

13) En Unix System V, Puede asignarse un bloque a un archivo sin acceder previamente al superblock?

Falso. Si o si tiene que consultar el superbloque para encontrar un bloque libre.

14) En Unix System V, al crear un archivo en un filesystem, indique qué se modifica:

-directorio al que pertenece

-superblock

-lista de i-nodos de todos los filesystems.

Directorio**:**	✅ Sí, se agrega la entrada del nuevo archivo.
Superbloque**:**	✅ Sí, se actualizan los contadores de inodos y bloques libres.
Lista de i-nodos de todos los filesystems:	❌ No, solo se modifica la lista de inodos del FS donde se crea el archivo.

15) En Unix System V, puedo crear un archivo en un filesystem no montado?

Falso. Para que un sistema de archivos sea accesible por el sistema operativo, primero debe estar montado en el árbol de directorios.

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

Falso. Cada sistema de archivos en un disco puede tener su propio tamaño de bloque definido al momento de su creación.

17) Cuando un archivo se borra, se ponen en cero los bloques

Falso. Simplemente se marcan como libres y quedan disponibles para ser reutilizados.

18) La estructura del filesystem define el tamaño máximo del archivo.

Verdadero. Está limitado por:

- El tamaño del **bloque**
- La cantidad de **punteros en el inodo**
- El esquema de **punteros directos e indirectos**
- El tamaño del **sistema de archivos**

19) La estructura del filesystem define la longitud máxima del nombre

Verdadero. La estructura del sistema de archivos en Unix System V define la longitud máxima del nombre de archivo, y este límite depende de la versión específica del sistema de archivos utilizado.

BUFFER CACHÉ

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

Verdadero. Esto ocurre cuando un proceso ha modificado el buffer y el sistema decide posponer la escritura en disco para mejorar el rendimiento.

21) El inodo de un archivo que se está usando debe estar en algún buffer del buffer cache.

Falso. El inodo de un archivo en uso debe estar en memoria (tabla de inodos en RAM), pero no necesariamente en el Buffer Cache.

22) Un buffer delayed write puede volver a estar ocupado, si lo pide un proceso, pero antes debe grabarse a disco. (importa el bloque en el buffer)

Falso.

- Si un proceso solo quiere leer/escribir el mismo bloque, puede reutilizar el buffer sin escribir en disco primero.
- Si el buffer se necesita para otro bloque diferente, entonces sí se debe escribir en disco antes de ser reutilizado.

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

Verdadero. Ya que contiene la información necesaria sobre los bloques libres del sistema de archivos.

24) Un proceso esperando por un buffer delayed write y ocupado, deberá esperar la escritura a disco antes de que se le asigne a él. (importa el bloque en el buffer)

Falso.

Depende del bloque en el buffer:

- Si es el mismo bloque, solo espera a que se libere.
- Si es otro bloque, debe esperar la escritura a disco antes de que se le asigne.

25) Las hash queues sirven para buscar por un buffer en particular

Verdadero. Las hash queues sirven para buscar un buffer en particular de manera eficiente dentro del buffer cache, reduciendo el tiempo de acceso a los bloques almacenados en memoria.

26) La free list sirve para buscar por cualquier buffer.

Falso. La free list en Unix System V no se usa para buscar cualquier buffer, sino únicamente aquellos que están disponibles para su reutilización.

27) No puede haber más de un proceso esperando por un buffer.

Falso. Sí, puede haber más de un proceso esperando por un buffer en Unix System V, especialmente si el buffer está ocupado o marcado como de escritura diferida. Los procesos deben esperar a que el buffer sea liberado antes de poder acceder a él.

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

Verdadero. Los datos se escriben en disco antes de que el buffer se ponga en la free list. Esto asegura que no se pierdan datos y que el sistema mantenga la coherencia entre la memoria y el disco.

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

Verdadero. Los datos se escriben en disco antes de que el buffer se ponga en la free list. Esto asegura que no se pierdan datos y que el sistema mantenga la coherencia entre la memoria y el disco.

30) Puede un buffer en la free list, estar ocupado?

Un buffer en la free list **NO** puede estar ocupado al mismo tiempo. Si un buffer está en la free list, significa que está disponible para su reutilización, es decir, no está siendo usado por ningún proceso.

31) Si un buffer está primero en la free list y en ese momento lo pide un proceso: ¿dónde va cuando se libere?

Cuando un buffer que estaba primero en la free list es solicitado por un proceso y luego liberado, **regresará a la free list** después de ser utilizado, después de haber sido escrito en disco si tenía escritura diferida.

32) Si un proceso necesita un buffer y la free list está vacía, el proceso se aborta.

Falso. No necesariamente. Si un proceso necesita un buffer y la free list está vacía, el proceso no se aborta inmediatamente en Unix System V. El sistema maneja esta situación de manera diferente, y existen mecanismos para gestionar la falta de buffers libres.

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

Verdadero. Si una hash queue está vacía, el sistema puede tomar un buffer de otra cola para cumplir con la solicitud de un proceso. Esto es parte del mecanismo de gestión eficiente de buffers en el buffer cache.

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

Falso. La tabla de páginas reside en la memoria principal, y el acceso a ella se realiza a través de la gestión de la memoria virtual, con el soporte de mecanismos como el TLB. El buffer cache se utiliza para el almacenamiento de bloques de datos del disco, no para la gestión directa de la memoria virtual.

35) ¿Qué conviene más? ¿Más cantidad de hash queues con pocos elementos, o menos cantidad de hash queues con más elementos?

- **Más colas pequeñas** suele ser beneficioso cuando hay un acceso **distribuido uniformemente** a los bloques de disco y se quiere minimizar el impacto de las colisiones.
- **Menos colas grandes** puede ser útil cuando hay una **alta concentración de accesos** en ciertas áreas y se prefiere una gestión más sencilla de las colas.

En general, el diseño óptimo depende del **patrón de acceso** y la **carga del sistema**. En muchos sistemas, el diseño más eficiente suele ser un **equilibrio** entre ambas opciones, es decir, tener un número **moderado de colas** que no sean ni demasiado grandes ni demasiado pequeñas.

AUTOEVALUACIÓN ACTUAL:

36) ¿Cuál técnica de reemplazo de páginas favorece al control de tasa de fallo de páginas de un proceso?

Reemplazo LOCAL.

37) En el diseño de la E/S no bloqueante, un proceso que realiza una operación continuará **su ejecución** por lo que deberá consultar el estado de la operación de forma explícita.

38) En el diseño de la E/S de un sistema operativo, se busca:

Unificar la forma de hacer E/S de para todos los dispositivos.

39) Los drivers de un dispositivo se ejecutan en modo kernel, aun cuando los mismos fueron diseñados por personas que no participan en el desarrollo del kernel de SO.

Verdadero.

40) Entre los servicios que brindan en el diseño de la E/S de un SO se encuentran la planificación de requerimientos y buffering de datos. En cambio, el manejo de errores de los dispositivos es delegado a los procesos de usuario que quieran utilizar dichos dispositivos.

Falso.

41) Suponiendo el manejo de espacio libre un disco por tabla de bits, la utilización de clusters de 8 sectores de disco hará que la tabla de bits sea más chica que si se utilizará clusters de 4 sectores de disco.

Verdadero.

42) Para poder crear un archivo "/var/log/milog", el SO necesita acceder a los I-NODOS de los directorios que participan en el path.

Verdadero.

43) Para la gestión de espacio de disco, el uso de clusters de gran tamaño podrá causar fragmentación:

INTERNA.

44) En la técnica de administración de archivo indexada, el tamaño máximo de un archivo lo determina:

La cantidad de índices definidos en la estructura de datos utilizada.

45) Afirmaciones sobre el FileSystem de Unix System V:

- Produce fragmentación externa. FALSO

- La cantidad de índices de sectores/clusters que se pueden utilizar por archivos es ilimitada. FALSO

- Todos los atributos de un archivo se encuentran en el I-Nodo. FALSO

- Produce fragmentación interna. FALSO

- La cantidad de índices de sectores/clusters que se pueden utilizar por archivos es ilimitada. VERDADERO

- El tamaño de la tabla de I-Nodos es dinámico. FALSO

46) Afirmaciones sobre si se modifica el nombre de un archivo el FileSystem de Unix System V:

- Se modifica el tamaño de la tabla de I-Nodos. FALSO

- Se modifica el I-Nodo del archivo al que se la cambió el nombre FALSO

- Se modifica el contenido del archivo al que se la cambió el nombre. FALSO

- Se modifica el contenido del directorio donde se encuentra ubicado el archivo. VERDADERO

- Se modifica el I-Nodo del directorio donde se encuentra ubicado el archivo. VERDADERO

47) Un filesystem es un componente de hardware que define el formato de los archivos, sus estructuras y brinda las operaciones necesarias para la manipulación de los mismos.

Falso.

48) En la administración de espacio de discos “Enlazada”, cada entrada de la FAT (para c/ archivo) contiene una lista con las referencias a todos los bloques de cada archivo.

Falso.

49) Qué tecnica de administración de espacio puede producir fragmentación externa:

CONTINUA.

50) En Buffer Cache, todos los buffers que se encuentran en la free list están vacíos (sin bloques de disco).

Falso.

51) En Buffer Cache, cuando un header en estado ocupado es liberado y el proceso que lo tenía había modificado el contenido del bloque es:

Colocado al final de la free list.

52) Suponiendo que en un SO se produce hiperpaginación. Disminuir el grado de multiprogramación ayudará a que mejore la productividad de la CPU.

Verdadero.

53) En administración de memoria con paginación donde se utiliza una Tabla de Páginas Invertida existe:

Una tabla invertida para todas las páginas que se encuentran cargadas en memoria principal.

54) Todo fallo de página que causa la ejecución de un proceso generará al menos dos operaciones de E/S: una escritura de una página en el área de intercambio y otra de la lectura de la página que corresponde a la dirección que generó el fallo en si.

Falso.

55) Si un proceso quiere modificar datos en un área compartida con otro proceso, puede hacerlo directamente si el otro proceso sólo accede en modo lectura.

Falso.

56) Para una gestión eficiente del área de intercambio (Swap Area) indique cual/es opciones deberán ser guardades en la misma:

- Páginas de:

LOS STACKS

CÓDIGO DEL KERNEL

DE DATOS (VARIABLES)

CÓDIGO DEL PROCESO

CÓDIGO DE LIBRERÍAS UTILIZADAS POR EL PROCESO

57) Cuando en la técnica de frecuencia de fallo de página todos los procesos se tienen una frecuencia mayor que el límite superior definido y no hay marcos libres nos encontramos con el problema de Hiperpaginación

Verdadero.

58) En Working Set, para aproximarse al modelo de localidad la selección de un delta CHICO **causará que en el conjunto de trabajo de un momento dado el proceso no cuente con toda su localidad.**

59) Afirmaciones sobre el Translation Lookaside Buffer (TLB):

- Si todas las entradas buscadas generan un TLB MISS, su uso perjudica al tiempo de resolución. VERDADERO

- Los cambios de contexto invalidan la TLB. VERDADERO

- Existe una única TLB y sus entradas son compartidas por todos los procesos. FALSO

- Sus entradas son administradas por el Kernel. FALSO

- Es una memoria caché que contiene las entradas de la tabla de páginas usadas recientemente. VERDADERO

- Para realizar una resolución de direcciones, primero se verifica en la tabla de páginas y si el bit V es 0, se analiza la TLB. FALSO

60) Siempre que la página buscada no se encuentre en la TLB (TLB MISS) se producirá fallo de página.

Falso.

61) Un tamaño de página GRANDE **resulta beneficioso para subir o bajar páginas del disco de la computadora.**

62) Luego de cargar una página a memoria por un fallo de página el responsable de poner el bit V en 1 es:

EL KERNEL

63) Dado un SO que se encuentra con hiperpaginación (Trashing), aumentar el grado de multiprogramación ayudará a que el problema se resuelva.

Falso.

64) En Working Set, la elección de un delta demasiado grande en un momento dado causará que los procesos no cuenten con todas las páginas necesarias.

Falso.