Repaso

Segundo Parcial Teórico

Memoria (2da parte), Archivos, Buffer Cache

1. La paginación por demanda es u na forma de implementar memoria virtual.
2. En la administración segmentada de memoria hay una tabla general para todo el sistema.
3. Para solucionar un page fault, habrá que hacer en algún momento un context switch.
4. La resolución de direcciones en el momento de la carga facilita la administración paginada.
5. Si un proceso contara con los frames que necesita, no tendría ningún page fault (falla de página).
6. La tabla de páginas es parte de: el contexto / la PCB / el espacio de direcciones / la pila del proceso
7. El tamaño de la tabla de segmentos depende del tamaño del proceso.
8. Si disminuyo la cantidad de bits del desplazamiento de una dirección de memoria, los frames serán más chicos.
9. La cantidad máxima de páginas en memoria depende sólo del tamaño del proceso.
10. EL bit de modificación relacionado con una página generará una I/O adicional.
11. Analice tamaños de página y page fault.
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.
14. Qué es el working set? Es siempre el mismo?
15. Qué consecuencias puede tener la hiperpaginación.
16. Que pasa si el delta elegido es muy chico? Y si es muy grande?
17. Diferencia entre reemplazo global y local.
18. Con el reemplazo local no cambia la cantidad de frames asignados al proceso
19. Diferencia entre asignación equitativa y proporcional.
20. Secuencia de resolución de page fault incluyendo TLB y cache común de memoria.
21. En cuanto estados del bit M y R: CUAL seria la página ideal para elegir como pagina victima?

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

.

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?

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

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

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

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

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

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.

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

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

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

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

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

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

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

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.

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

44. La free list sirve para buscar por cualquier buffer.
45. Los buffers del buffer cache pueden tener distinto tamaño.

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

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

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

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

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

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

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

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

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